НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК 004.31

«До захисту допущено»

Завідувач кафедри СПСКС

_____ В.П.Тарасенко_
(підпис) (ініціали, прізвище)

"___" _____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія (Спеціалізовані комп'ютерні системи)
на тему: «Методи захисту спеціалізованих моніторингових комп'ютерних засобів на ПЛІС»

Виконав: студент ІІ курсу, групи КВ-63м
(шифр групи)

Тарасенко Георгій Олегович
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник к.т.н., доцент Клятченко Ярослав Михайлович
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент д.т.н., професор Симоненко Валерій Павлович
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
Спеціалізовані комп'ютерні системи

ЗАТВЕРДЖУЮ
Завідувач кафедри СПСКС
_____
_В.П.Тарасенко__
(підпис) (ініціали, прізвище)
«___»_____2018р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Тарасенка Георгія Олеговича
(прізвище, ім'я, по батькові)

1. Тема дисертації «Методи захисту спеціалізованих моніторингових комп'ютерних засобів на ПЛІС»,
науковий керівник дисертації Клятченко Ярослав Михайлович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «22» березня 2018 р. №986-с

2. Термін подання студентом дисертації 11 травня 2018 р.

3. Об'єкт дослідження – процеси, що впливають на ступінь захищеності спеціалізованих моніторингових обчислювальних засобів на базі програмовних інтегральних середовищ від атак та втручань.

4. Предмет дослідження – методи забезпечення захисту спеціалізованих моніторингових комп'ютерних засобів на ПЛІС.

5. Перелік завдань, які потрібно розробити

- дослідження впливу загроз та атак на блоки інтелектуальної власності проектів спеціалізованих обчислювальних засобів на ПЛІС- оцінка ефективності існуючих методів захисту IP-блоків в проектах на ПЛІС із урахуванням архітектурно-структурної організації цих напівфабрикатів;
- розробка нових методів для підвищення рівня захищеності блоків інтелектуальної власності проектів апаратних засобів на ПЛІС;
- оцінка рівня захищеності IP-блоків від атак та загроз різних типів.

6.Перелік ілюстративного матеріалу

- Графік результатів синтезу.
- Графік результатів синтезу після проведення обфускації.
- Реалізація методу MV на ядрах UART.
- Реалізація методу MCRC на ядрах ALU.
- Реалізація методу MVR на ядрах AES.
- Блок-схема алгоритму мультиплексування даних із виходів реконфігурованих IP та реалізація схеми виявлення троянців .
- Мультиплексування виходів IP зі змінюваною конфігурацією для вибору вірного і додавання частини CRC для перевірки виходу.
- Схема голосування CRC.

7.Перелік публікацій

- IX науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2017 (Київ, 19-21 квітня 2017 р.);
- International Conference on Computer Science, Engineering and Education Applications (ICCSEEA2018);
- X науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2018 (Київ, 21-23 квітня 2018 р.).

8. Дата видачі завдання 5 вересня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка

1	Ґрунтовне ознайомлення з предметною областю дослідження	10.10.2017	
2	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури	12.12.2017	
3	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	16.01.2018	
4	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	14.02.2018	
5	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	15.03.2018	
6	Проведення наукового дослідження; робота над третім розділом магістерської дисертації; підготовка матеріалів доповіді на конференції ПМК-2018.	20.02.2018	
7	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу;	16.04.2018	
8	Оформлення текстової і графічної частини магістерської дисертації	20.04.2018	
9	Попередній розгляд магістерської дисертації на кафедрі	26.04.2018	

Студент _____ Г.О.Тарасенко
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Я.М.Клятченко
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Актуальність. Забезпечення заданого рівня надійності і безпеки використання об'єктів критичного застосування (аерокосмічні комплекси, підприємства енергетичної галузі, підприємства із небезпечним виробництвом, об’єкти транспорту) залежить від ступеня безвідмовності програмно-апаратних компонент, досконалості архітектурних рішень, використовуваних методів і засобів стійкості до відмов, що викликані різними факторами (фізичними і проектними дефектами, дефектами взаємодії із зовнішнім середовищем). Застосування ефективних методів, моделей та спеціальних архітектурно-структурних рішень в системах моніторингу стану критичних об’єктів дозволяє вирішити ряд проблем завдяки, в тому числі, і застосування програмовних логічних інтегральних середовищ (ПЛІС) для практичної реалізації таких систем. При проектуванні сучасних систем на кристалі широко використовуються готові модулі - блоки інтелектуальної власності: IP, IP-блоки або IP-core (intellectual property), що втілені у функціональних можливостях пристроїв та які реалізують досить складні алгоритми перетворення і обробки даних. Однак, багатий досвід використання ПЛІС сучасного рівня складності та сенсорних мереж в критичних додатках, що накопичений в Україні, вимагає більш детального осмислення і аналізу, а таке дослідження повинно проводитися із урахуванням можливих ризиків, які мають місце при використанні ПЛІС-технологій із дотриманням вже відомих базових принципів забезпечення надійності, функціональної та інформаційної безпеки. Крім того, самі пристрої на ПЛІС та мережі сенсорів мають можливість змінювати свою конфігурацію, що вимагає реалізації гнучких засобів їхньої інтеграції. Саме тому дослідження, спрямовані на пошук шляхів підвищення рівня захищеності систем моніторингу вистану об’єктів критичного застосування

Об’єктом дослідження є процеси, що впливають на ступінь захищеності спеціалізованих моніторингових обчислювальних засобів на базі програмовних інтегральних середовищ від атак та втручань.

Предметом дослідження є методи забезпечення захисту спеціалізованих моніторингових комп’ютерних засобів на ПЛІС.

Мета і задачі дослідження. Метою дослідження є підвищення рівня захищеності спеціалізованих обчислювальних засобів для моніторингу об’єктів критичного застосування та перевірка ефективності запропонованих структур засобів захисту.

Для досягнення поставленої мети в роботі вирішуються наступні задачі:

- дослідження впливу загроз та атак на блоки інтелектуальної власності проектів спеціалізованих обчислювальних засобів на ПЛІС ;

- оцінка ефективності існуючих методів захисту IP-блоків в проектах на ПЛІС із урахуванням архітектурно-структурної організації цих напівфабрикатів;

- розробка нових методів для підвищення рівня захищенності блоків інтелектуальної власності проектів апаратних засобів на ПЛІС;

- оцінка рівня захищеності IP-блоків від атак та загроз різних типів.

Методи дослідження базуються на використанні методів теорії ймовірностей та математичного статистичного аналізу, комп'ютерної схемотехніки, теорії організації обчислювальних процесів для створення спеціалізованих комп'ютерних систем на ПЛІС.

Наукова новизна одержаних результатів полягає в тому, що пропонуються новий, комплексний метод захисту блоків інтелектуальної власності спеціалізованих обчислювальних комп'ютерних систем на ПЛІС для моніторингу об'єктів критичного застосування та засоби, які реалізують положення цих методів.

Практична цінність запропонованого методу захисту блоків інтелектуальної власності в спеціалізованих обчислювальних комп'ютерних систем на ПЛІС полягає у збільшенні рівня захищеності IP-блоків від атак та загроз, що забезпечує підвищення ефективності застосування таких блоків в апаратних засобів для моніторингу об'єктів критичного застосування.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювались на:

- IX науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2017 (Київ, 19-21 квітня 2017 р.);
- International Conference on Computer Science, Engineering and Education Applications (ICCSEEA2018);
- X науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2018 (Київ, 21-23 квітня 2018 р.).

Публікації. Результати роботи опубліковані в 4 наукових працях, з яких 2 – тези доповідей.

Структура і об'єм роботи. Магістерська дисертація складається з вступу, трьох розділів, висновків та додатків.

У вступі розглянуто загальну характеристику роботи, виконано оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень.

У *першому* розділі проаналізовано особливості архітектурно-структурної організації напівфабрикатів ПЛІС. Проведено класифікацію та оцінено рівень негативного впливу внаслідок атак та загроз на блоки інтелектуальної власності, які використовуються для реалізації комп'ютерних засобів.

У *другому* розділі проведено огляд та порівняння засобів захисту блоків інтелектуальної власності комп'ютерних засобів на ПЛІС.

У *третьому* розділі розгорнуто ідеї на яких базується комплексний метод захисту проектів спеціалізованих моніторингових засобів шляхом застосування спеціальних апаратних засобів для захисту проекту на ПЛІС. Показано способи підвищення ефективності цієї схеми за рахунок часткового реконфігорування та введення криптографічних примітивів і персоналізації. Проведено оцінку ефективності запропонованого підходу та визначено ступінь захищеності IP-блоків.

У *висновках* показано основні результати роботи. Визначено умови для яких розроблений метод захисту є ефективним. Також запропоновано шляхи покращення розробленого метода та рекомендації щодо подальших досліджень.

У *додатках* подано копії графічних матеріалів, лістинги програм, копії публікацій.

Ключові слова: Intellectual property core, programmable logical devices, bitstream encryption, FPGA design protection, AES, HMAC, ECC, CRC, SEU.

РЕФЕРАТ

Актуальность. Обеспечение заданного уровня надежности и безопасности использования объектов критического применения (аэрокосмические комплексы, предприятия энергетической отрасли, предприятия с опасным производством, объекты транспорта) зависит от степени безотказности программно-аппаратных компонент, совершенства архитектурных решений, используемых методов и средств отказоустойчивости, вызванных различными факторами (физическими и проектными дефектами, дефектами взаимодействия с внешней средой). Применение эффективных методов, моделей и специальных архитектурно-структурных решений в системах мониторинга состояния критических объектов позволяет решить ряд проблем благодаря, в том числе и применение программируемых логических интегральных сред (ПЛИС) для практической реализации таких систем. При проектировании современных систем на кристалле широко используются готовые модули – блоки интеллектуальной собственности: ИС, IP-блок или IP-core (intellectual property), которые воплощены в функциональных возможностях устройств и которые реализуют достаточно сложные алгоритмы превращения і обработки данных. Однако, богатый опыт использования ПЛИС современного уровня сложности и сенсорных сетей в критических приложениях, накопленный в Украине, требует более детального осмысления и анализа, а такое исследование должно проводиться с учетом возможных рисков, которые имеют место при использовании ПЛИС-технологий с соблюдением уже известных базовых принципов обеспечения надежности, функциональной и информационной безопасности. Кроме того, сами устройства на ПЛИС и сети сенсоров имеют возможность изменять свою конфигурацию, требует реализации гибких средств ихнои интеграции. Именно поэтому исследования, направленные на поиск путей повышения уровня защищенности систем мониторинга состояния объектов критического применения

Объектом исследования являются процессы, влияющие на степень защищенности специализированных мониторинговых вычислительных средств на базе программируемых интегральных сред от атак и вмешательств.

Предметом исследования являются методы обеспечения защиты специализированных мониторинговых компьютерных средств на ПЛИС.

Цель и задачи исследования. Целью исследования является повышение уровня защищенности специализированных вычислительных средств для мониторинга объектов критического применения и проверка эффективности предложенных структур средств защиты.

Для достижения поставленной цели в работе решаются следующие задачи:

- исследование влияния угроз и атак на блоки интеллектуальной собственности проектов специализированных вычислительных средств на ПЛИС;
- оценка эффективности существующих методов защиты IP-блоков в проектах на ПЛИС с учетом архитектурно-структурной организации этих полуфабрикатов;
- разработка новых методов для повышения уровня защищенности блоков интеллектуальной собственности проектов аппаратных средств на ПЛИС;
- оценка уровня защищенности IP-блоков от атак и угроз различных типов.

Методы исследования базируются на использовании методов теории вероятностей и математического статистического анализа, компьютерной схемотехники, теории организации вычислительных процессов для создания специализированных компьютерных систем на ПЛИС.

Научная новизна исследования заключается в том, что предлагаются новый, комплексный метод защиты блоков интеллектуальной собственности специализированных вычислительных компьютерных систем на ПЛИС для мониторинга объектов критического применения и средства, которые реализуют положения этих методов.

Практическая ценность предложенного метода защиты блоков интеллектуальной собственности в специализированных вычислительных компьютерных систем на ПЛИС заключается в увеличении уровня защищенности IP-блоков от атак и угроз, обеспечивает повышение эффективности применения таких блоков в аппаратных средств для мониторинга объектов критического применения.

Апробация работы. Основные положения и результаты работы были представлены и обсуждались на:

- IX научной конференции магистрантов и аспирантов «Прикладная математика и компьютинг» ПМК-2017 (Киев, 19-21 апреля 2017)
- International Conference on Computer Science, Engineering and Education Applications (ICCSEEA2018)
- X научной конференции магистрантов и аспирантов «Прикладная математика и компьютинг» ПМК-2018 (Киев, 21-23 апреля 2018).

Публикации. Результаты работы опубликованы в 4 научных работах, из которых 2 - тезисы докладов.

Структура и объем работы. Магистерская диссертация состоит из введения, трех глав, заключения и приложений.

Во введении рассмотрена общая характеристика работы, выполнена оценка современного состояния проблемы, обоснована актуальность направления исследований.

В первой главе проанализированы особенности архитектурно-структурной организации полуфабрикатов ПЛИС. Проведена классификация и оценен уровень негативного воздействия в результате атак и угроз на блоки интеллектуальной собственности, используемых для реализации компьютерных средств.

Во второй главе проведен обзор и сравнение средств защиты блоков интеллектуальной собственности компьютерных средств на ПЛИС.

В третьем разделе развернуты идеи на которых базируется комплексный метод защиты проектов специализированных мониторинговых средств путем применения специальных аппаратных средств для защиты проекта на ПЛИС. Показаны способы повышения эффективности этой схемы за счет частичной реконфигурац ии и введение криптографических примитивов и персонализации. Проведена оценка эффективности предложенного подхода и определена степень защищенности IP-блоков.

В выводах показаны основные результаты работы. Определены условия для которых разработан метод защиты является эффективным. Также предложены пути улучшения разработанного метода и рекомендации по дальнейшим исследованиям.

В приложениях представлены копии графических материалов, листинги программ, копии публикаций.

Ключевые слова: Intellectual property core, programmable logical devices, bitstream encryption, FPGA design protection, AES, HMAC, ECC, CRC, SEU.

ABSTRACT

Theme topicality. Provision of a certain level of reliability and safety of the use of objects of critical application (aerospace complexes, enterprises of the energy industry, enterprises with hazardous production, objects of transport) depends on the degree of failures of software and hardware components, the perfection of architectural solutions, methods and means of resistance to failures , caused by various factors (physical and design defects, defects in interaction with the external environment). The application of effective methods, models and special architectural and structural solutions in critical-object monitoring systems allows us to solve a number of problems, including the use of programmable logical integral environments (FPGAs) for the practical implementation of such systems. The following are examples of the training system in the form of the following examples: intellectual property units: IPs, IP-blocks of either IP-core (intellectual property) embodied in the functional capabilities of the devices, and which translate into specific algorithms for petevoping and decompression. However, the rich experience of using the current level of complexity and sensor networks in critical applications in Ukraine, which has accumulated in Ukraine, requires more detailed reflection and analysis, and such research should be conducted taking into account the potential risks that occur when using FPGA technologies with the observance of already known basic principles of reliability, functional and information security. In addition, the devices themselves on the FPGA and network sensors have the ability to change their configuration, which requires the implementation of flexible means of their integration. That is why the research aimed at finding ways to increase the level of security of monitoring systems for the critical use objects

The object of the research is the processes that affect the degree of protection of specialized monitoring computers based on programmable integral environments from attacks and interventions.

The subject of the research is the methods of ensuring the protection of specialized monitoring computer tools in the FPGA.

The purpose and tasks of the study. The purpose of the study is to increase the level of protection of specialized computing facilities for monitoring critical objects and to check the effectiveness of the proposed structures of protection.

To achieve this goal, the following tasks are solved:

- investigation of the impact of threats and attacks on the blocks of intellectual property of specialized computer software projects on the FPGA;

- evaluation of the effectiveness of existing methods of protection of IP blocks in projects on the FPGA, taking into account the architectural and structural organization of these semi-finished products;

- development of new methods for increasing the level of protection of intellectual property units of hardware projects on FPGA;

- assessment of the level of protection of IP blocks from attacks and threats of various types.

The research methods are based on the use of methods of probability theory and mathematical statistical analysis, computer circuitry, the theory of the organization of computational processes for the creation of specialized computer systems on the FPGA.

The scientific novelty of the obtained results is that a new, integrated method of protection of intellectual property blocks of specialized computer systems on FPGA is offered for monitoring of critical objects and means that implement the provisions of these methods.

The practical value of the proposed method for protecting blocks of intellectual property in specialized computing computer systems in the FPGA is to increase the level of protection of IP blocks from attacks and threats, which ensures an increase in the efficiency of the use of such blocks in hardware for monitoring critical use objects.

Test work. The main provisions and results of work were presented and discussed at:

- IX scientific conference of masters and postgraduates "Applied Mathematics and Computer", PMK-2017 (Kyiv, April 19-21, 2017);

- International Conference on Computer Science, Engineering and Education Applications (ICCSEEA2018);

- X scientific conference of masters and postgraduates "Applied Mathematics and Computer", PMK-2018 (Kyiv, April 21-23, 2018).

Publications The results of the work are published in 4 scientific works, of which 2 are theses of reports.

Structure and volume of work. The master's dissertation consists of an introduction, three sections, conclusions and appendices.

In the introduction a general description of the work is considered, the estimation of the current state of the problem is made, the relevance of the research direction is substantiated.

In the first section the features of the architectural and structural organization of semi-finished FPGA are analyzed. The classification and evaluation of the level of negative influence as a result of attacks and threats to the blocks of intellectual property, which are used for the implementation of computer tools.

In the second section, a review and comparison of the means of protection of intellectual property units of computer tools on the FPGA.

The third section deploys ideas on which the integrated method of protecting projects of specialized monitoring tools is based on the use of special hardware for the protection of the project on the FPGA. The ways of increasing the efficiency of this scheme by means of partial reconfiguration and introduction of cryptographic primitives and personalization are shown. The effectiveness of the proposed approach was evaluated and the degree of protection of IP blocks was determined.

The conclusions show the main results of work. Defined conditions for which the developed method of protection is effective. Ways to improve the developed method and recommendations for further research are also suggested.

Applications contain copies of graphic materials, program listings, copies of publications.

Keywords: Intellectual property core, programmable logical devices, bitstream encryption, FPGA design protection, AES, HMAC, ECC, CRC, SEU.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ

ВІС	Велика інтегральна схема
НВІС	Надвелика інтегральна схема
ПЗ	Програмне забезпечення
ПЛІС	Програмовна логічна інтегральна схема
САПР	Система автоматизованого проектування
ASIC	Application-specific integrated circuit (замовна інтегральна схема)
ASSP	Application-specific standart product (замовна спеціалізована інтегральна схема)
CLB	Configurable logic block (логічний блок, що конфігурується)
CPLD	Complex programmable logic device (складний програмовний логічний пристрій)
CPU	Central processing units (центральний процесор)
FPGA	Field-programmable gate array (програмована користувачем вентильна матриця)
IP	Intellectual property (блок інтелектуальної власності)
LUT	Look-Up table (логична таблиця)
NoC	Network-on-a-Chip (мережа на кристалі)
PLD	Programmable logic device (програмовний логічний пристрій)
RTL	Register-transfer level (рівень регістрових передач)
SoC	System-on-a-Chip (система на кристалі)
VHDL	VHSIC hardware description language (мова опису апаратних засобів)

ВСТУП

Однією із найважливіших наукових і практичних задач комп'ютерної інженерії є забезпечення надійної роботи програмних, апаратних та мережевих компонентів і систем. Це пов'язано із зростанням їхнього впливу на надійність і безпеку комп'ютеризованих комплексів об'єктів критичного застосування (аерокосмічні комплекси, підприємства енергетичної галузі, підприємства із небезпечним виробництвом, об'кти транспорту). Також, неможливо вирішувати задачу безвідмовності таких систем без мониторінгу середовища, в якому вони функціонують та наслідків відмов. Необхідність досліджень в даній області обумовлюється такими факторами як масштабність різних типів комп'ютерних і телекомунікаційних систем; невизначеність і змінність характеристик їх компонентів; розподіленість архітектури та динамізм процесів функціонування; агресивність середовища і розширення переліку причин, що призводять до повного або часткового невиконання передбачених специфікацією функцій внаслідок дефектів їхніх апаратних, програмних або мережевих засобів, помилок персоналу, а особливо, інформаційних атак та загроз від дій зловмисників.

Забезпечення заданого рівня надійності і безпеки використання об'єктів критичного застосування залежить від рівня безвідмовності апаратних компонент, досконалості архітектурних рішень, використовуваних методів і засобів стійкості до відмов, що викликані різними факторами. Застосування ефективних методів та спеціальних архітектурно-структурних рішень в системах моніторингу стану критичних об'єктів дозволяє вирішити ряд проблем завдяки, в тому числі, і застосування програмовних логічних інтегральних схем (середовищ) (ПЛІС) для практичної реалізації таких систем. Однак, багатий досвід використання ПЛІС сучасного рівня складності вимагає більш детального дослідження загроз та врахування можливих ризиків, які мають місце при використанні цих ПЛІС-технологій із дотриманням вже відомих базових принципів забезпечення надійності, функціональної та інформаційної безпеки. Крім того, самі пристрої на ПЛІС мають можливість змінювати свою конфігурацію, що дозволяє реалізувати гнучкі засоби їх інтеграції. Саме тому дане дослідження спрямовано на пошук шляхів підвищення ефективності

функціонування спеціалізованих систем на ПЛІС для моніторингу об’єктів критичного застосування.

РОЗДІЛ 1

АНАЛІЗ АРХІТЕКТУРНО-СТРУКТУРНОЇ ОРГАНІЗАЦІЇ СУЧАСНИХ ПЛІС ТА КЛАСИФІКАЦІЯ ІСНУЮЧИХ МЕТОДІВ ТА ЗАСОБІВ ЗАХИСТУ ПРОЕКТІВ КОМП'ЮТЕРНИХ СИСТЕМ НА ЇХ ОСНОВІ

1.1. Сучасні форми використання пристроїв програмовної логіки в проектах комп'ютерних засобів для моніторингу об'єктів критичного застосування

Розширення сфер застосування традиційних комп'ютерних систем, створення потужних пошукових систем та інше, обумовлюють необхідність реалізацій операційними частинами обчислювальних засобів спеціалізованих операцій для обробки масивів, матриць, векторів тощо, що базуються на елементарних операціях. Такі задачі як, моделювання траєкторій об'єктів або іх сукупності, обробка сигналів та зображень у реальному часі, складна реалістична комп'ютерна графіка, online-обробка пошукових запитів пов'язані із швидкісним (багаторазовим) виконанням до декількох мільярдів елементарних операцій за секунду. Послідовне виконання великої кількості елементарних операцій займає відносно великий час, а тому навіть прості задачі, а також задачі, які мають вирішуватися в реальному часі, викликають необхідність застосування спеціалізованих прискорювачів [1]. Для створення цих апаратних прискорювачів можна у вигляді замовних BIC/HBIC (ASIC), що не завжди є доцільним – вартість підготовки до виробництва виробу не завжди може окупитися. Це унеможливлює та нівелює практичне застосування такого підходу для спеціалізованих обчислювальних засобів, які виготовляються невеликими тиражами. Як можливий варіант подолання цих недоліків деякий час тому виникла тенденція спільного використання обчислювальних систем на базі мікропроцесора і ПЛІС, що змогло задовольнити вимоги

широкого кола користувачів та започаткувати нові архітектурні форми ефективних обчислювальних засобів. Крім того, ПЛІС дозволяють за допомогою використання тільки персонального комп'ютера та системи автоматизованого проектування (САПР) ПЛІС реалізувати в кристалі практично будь-який проект цифрового пристрою за короткий проміжок часу. Застосування сучасних напівфабрикатів ПЛІС, які за характеристиками інтеграції, енергоспоживання, продуктивності досягли рівня традиційних BIC/HBIC, в сучасних умовах високої конкуренції на ринку та складних економічних умовах для багатьох розробників стає єдиним можливим рішенням на шляху реалізації проекту обчислювальної системи у цілому. Згідно [2], кількість початкових проектів цифрових пристроїв на ASIC йде на спад. Так, якщо в 2014 році 41% початкових проектів базувались на ASIC, то на 2016 рік їхня частка складала 34%. За самими оптимістичними сподіваннями [3] до 2019 року доля ASIC в початкових проектах має тенденцію до зниження зі швидкістю 3,8% на рік, і, відповідно зростання долі мікросхем програмованої логіки по створенню нових проектів має тенденцію до зростання.

Для розробників сучасних високоефективних цифрових пристроїв існує декілька напрямків реалізації проекту на базі ПЛІС:

- універсальна система – процесорна система на ПЛІС із спеціалізованим програмним забезпеченням (ПЗ). Для цього випадку можливе використання адаптованого ПЗ, що розроблено раніше. Даний напрямок доцільно використовувати для реалізації нескладних алгоритмів або для додатків із низькою інтенсивністю потоків даних. Перевагами такого рішення є незалежність від складності алгоритму та одноразовість витрат, що складають витрати на апаратну реалізацію на ПЛІС;
- спеціалізована система - спеціалізоване апаратне забезпечення з максимальною ефективністю реалізації конкретного алгоритму. Даний

варіант використовується для задач реального часу із інтенсивними потоками даних, які вимагають значних затрат на їхню реалізацію програмним методом з використанням універсальних обчислювальних засобів;

- комбінована система – поєднання традиційних обчислювальних засобів із спеціалізованим апаратним забезпеченням на ПЛІС.

Із застосуванням таких підходів, відомі використання ПЛІС при побудові [4-10]:

- апаратури спеціалізованого призначення (реалізація складних алгоритмів та операцій, апаратні прискорювачі обчислень, нетрадиційна обробка сигналів в реальному часі, створення засобів моніторингу та керування виробничими процесами, кодування/декодування інформації в телекомунікаційних системах, системи мережевого моніторингу з підвищеними вимогами до швидкодії);
- інтерфейсних засобів сполучення з апаратурою систем управління;
- засобів захисту інформації (контроль доступу до інформації на твердих магнітних і лазерних дисках — так звані індивідуальні електронні ключі доступу, шифрування інформації перед записом на диск або передачею в мережу);
- графічних прискорювачів для відтворення реалістичної графіки та обробки зображень;
- реалізація комунікаційних протоколів нижніх рівнів (такі задачі традиційно вирішуються з використанням апаратних засобів для формування бітових потоків і їхнього первинного декодування, встановлення з'єднань абонентів, тощо).

Останні роки для вирішення задач моніторингу об’єктів критичного засторсування все більшого застосування набувають так звані розподілені сенсорні мережі, які складаються з великої кількості недорогих та доступних

в серійному виробництві сенсорних комп’ютерних пристроїв, які мають відносно невелику точність, але здатні, працюючи як система, давати результати щодо оцінки стану об’єктів навколишнього середовища з необхідною точністю. Як правило, критичні об’єкти займають значний простір (зокрема, промислові підприємства, розташування військ на місцевості тощо), тому спостереження змін стану таких об’єктів в реальних умовах можливо лише завдяки використанню мереж сенсорів. Безперервний потік даних від великої кількості подібних пристроїв породжує наявність експоненційно зростаючого потоку інформації, що потребує подальшої обробки в реальному часі: фільтрації, агрегації індексації тощо. Наявні на сьогодні комп’ютерні засоби на основі ПЛІС дозволяють досягнути вирішення цих задач як з точки зору економічної доцільності, так і ефективності обробки надвеликих обсягів потокової інформації.

Розглянемо властивості та нові архітектурні особливості сучасних ПЛІС. Властивість репрограмування (реконфігурування) ПЛІС [11-16] – така властивість, що дає можливість переналаштування структури та функцій цифрового операційного засобу з метою підвищення ефективності (продуктивності, надійності, тощо) функціонування. На сьогоднішній день ПЛІС, що репрограмуються, забезпечують нові можливості обчислювальних систем, які побудовані на їхній основі. В класичному випадку можливість реконфігурування системи на ПЛІС відбувається на завершальній стадії розробки проекту. Але сучасні засоби проектування та конструкція мікросхем програмованої логіки дозволяють виконувати повну чи часткову реконфігурацію в процесі роботи системи. Маючи в своєму розпорядженні плату, що містить одну або декілька ПЛІС, і підключаючи її до комп'ютера стандартними засобами, можна отримати кілька різних спеціалізованих пристроїв. Досить завантажити в мікросхему одну з можливих конфігурацій із числа, тих що зберігаються на твердому диску комп'ютера, щоб викликати до виконання той або інший спеціалізований алгоритм обробки даних.

Користувач, також, може самостійно модифікувати роботу цього, пристрою, забезпечуючи реалізацію тих чи інших специфічних функцій, що не передбачені заздалегідь [17-18].

Однією із сучасних тенденцій у використанні ПЛІС є створення таких форм цифрових пристроїв як, наприклад, система на кристалі – SoC (System-on-a-chip), мережа на одному кристалі – NoC (Network-on-a-chip) та інші. SoC [19-21] – це спеціалізований цифровий виріб, конструктивно виконаний на кремнієвій підкладці, що характеризується низьким енергоспоживанням, малими розмірами, високою швидкодією і має наступні основні компоненти: процесор, пам'ять, логіку, інтерфейс вводу-виводу, перетворювачі інформації, шини обміну даними та інфраструктуру сервісного обслуговування. Функціональні блоки SoC орієнтовані на ефективне вирішення спеціалізованої задачі на відміну від мікроконтролера, що представляє собою універсальний обчислювач. Перевагами SoC є прозорість щодо наявності вбудованих компонентів, висока швидкодія при низькому енергоспоживанні, мініатюрні розміри, ефективне використання різних типів пам'яті, висока надійність, низька вартість готового виробу та високий коефіцієнт використання функціональностей. До недоліків SoC можна віднести значний час тестування та верифікації проекту, висока вартість розробки та макетування, складність відлагодження та виробництва, складність інтеграція блоків інтелектуальної власності [22] з різних джерел.

До однієї із еволюційних форм SoC можна віднести мережу на кристалі (NoC). NoC це спеціалізований мультиядерний цифровий виріб конвеєрного типу, що має властивості системи на кристалі та доповнений інтелектуальною структурою комунікацій, що може підтримувати різні архітектури і рівні протоколів обміну даними. Як ядро може застосовуватися Core (процесор або блок пам'яті), які організуються в масштабовані сегменти. Підтримка всіх рівнів протоколу обміну даними забезпечує високу швидкодію і пропускну здатність інформаційних каналів. Наявність

інтелектуальних мультиядерної архітектури і комутаторів забезпечує високий рівень розпаралелювання обчислювальних процесів та вводу/виводу даних [19].

Використання передових технологій в процесі виробництва ПЛІС-мікросхем призвело до значного розширення їх функціональних можливостей У першу чергу ця тенденція стосується ПЛІС з архітектурою FPGA. Внаслідок цього, в процесі проектування цифрових пристроїв ПЛІС може розглядатися як деяке мікропроцесорне ядро, що оточене вільними логічними ресурсами, які конфігуруються. За способом реалізації процесорні ядра, що використовуються при проектуванні сучасних обчислювальних засобів на основі ПЛІС, поділяють на програмні (soft-процесори, soft-ядра) та апаратні (hard processor) [23]. Програмні ядра або ядра, що конфігуруються, формуються на основі стандартних логічних ресурсів кристалів. Перевагами програмних процесорних ядер у порівнянні з апаратними є висока гнучкість, низька собівартість, відносно невеликий обсяг використовуваних ресурсів кристалів, можливість застосування в проектах, що реалізуються на базі найпоширеніших і доступних сімейств ПЛІС. Апаратні процесори - мікропроцесорні ядра, які виконані у вигляді відповідних інтегрованих апаратних блоків ПЛІС. Головна перевага апаратних мікропроцесорних ядер полягає в можливості функціонування з високими тактовими частотами. Наслідком цього є більш висока швидкодія у порівнянні з мікропроцесорними ядрами, що конфігуруються. Недоліками апаратних ядер є обмежена кількість моделей кристалів ПЛІС, в яких вони застосовуються, та їхня висока вартість. Але кількість початкових проектів на ПЛІС, що використовують процесорні ядра збільшується з кожним роком, в т.ч. за рахунок використання багатопроцесорної організації (розпаралелювання обчислювальних процесів), розширення сфер застосування ПЛІС, поступового витіснення середовищами з програмованою структурою традиційних ASIC тощо.

Однією із останніх тендецій є розвиток класу пристроїв-напівфабрикатів, як, наприклад пропонує фірма Xilinx, „All programmable devices”. Така мікросхема містить апаратне процесорне ядро, пам’ять, периферійні пристрої, вбудовані інтерфейси та матрицю програмованих логічних комірок, аналогічну до FPGA [24].

Продукція лідера ринку - компанії Xilinx охоплює найбільший сегмент промисловості програмованої логіки з 1984 року як за кількістю так і за різноманітністю модельного ряду продукції, засобів проектування, IP тощо. Для роботи із мікросхемами компанія Xilinx надає програмні засоби для реалізації цифрових пристроїв, розробки процесорних систем на базі напівфабрикатів програмованої логіки, а також засоби для відлагодження та підвищення продуктивності. Крім, властиво, розробки мікросхем, компанія Xilinx приділяє велику увагу засобам цифрової обробки сигналів та розробці різноманітних IP-ядер.

На даний час основна лінійка мікросхем програмованої логіки складається з пристроїв Серії 7 (Artix-7, Kintex-7, Virtex-7) [25], ПЛІС попередніх серій - Virtex-6 та Spartan-6 [26], мікросхем для автомобільної промисловості (XA Spartan-6/3A/3E) [27-28] та лінійки військового призначення (Artix-7Q, Kintex-7Q, Virtex-7Q/6Q/5Q/4Q, Spartan-6Q) [29]. Також Xilinx продовжує вдосконалювати мікросхеми архітектури типу CPLD (CoolRunner-II і XC9500XL) [30]. Нещодавно компанія представила нове покоління пристроїв що відносяться до класу програмованих SoC - Zynq-7000 [31].

Компанія Altera є основним конкурентом компанії Xilinx, по всіх основних напрямках діяльності. Перший із них - це виробництво ПЛІС як типів FPGA, та типу CPLD. Нещодавно Altera запропонувала нове сімейство із серії Stratix високопродуктивних мікросхем типу FPGA - Stratix 10 [32], що базується на новій технології виробництва (14 нм). Для менш ресурсомістких завдань компанія Altera пропонує серію ПЛІС архітектури

FPGA – Cyclone V [33]. А як компроміс між продуктивними Stratix і недорогими Cyclone - серію Arria [34], що має невелику вартість та низьке енергоспоживання, орієнтовану на використання як прийомопередавачів (трансіверів) комунікаційних систем. Для мобільних пристроїв випускається серія Max [35] на основі ПЛІС типу CPLD. Також, ПЛІС серій Cyclone та Arria випускаються як SoC із вбудованими апаратними ядрами архітектури ARM.

Починаючи із серії Stratix III у ПЛІС фірми Altera використовується технологія Programmable Power Technology, що дозволяє варіювати режим роботи та споживану потужність логічних комірок і в залежності від необхідної швидкодії. Мікросхеми компанії Altera активно застосовуються в багатьох областях, наприклад, в галузі комунікацій, у військових технологіях, в області телебачення, а також у різних мобільних пристроях.

Компанія Actel є відомим виробником невеликих і недорогих мікросхем типу FPGA, головною якістю яких є висока надійність. В 2010році Actel було придбано фірмою MicroSemi. У цілому мікросхеми, що випускаються компанією MicroSemi (Actel), за способом програмування можна розділити на два типи: с використанням flash-пам'яті та з одноразово програмованою пам'яттю (antifuse технологія). Обидва типи мікросхем забезпечують високий рівень захищеності інформації як від несанкціонованого доступу, так і від радіаційного випромінювання. Великою перевагою цих мікросхем є те, що не потрібно завантажувати конфігураційну послідовність при включенні живлення. Це означає, що вони готові до роботи відразу ж після запуску, і тому основними клієнтами MicroSemi(Actel) є компанії-виробники портативних пристроїв та автомобільна промисловість. Завдяки властивостям високої надійності та миттєвої готовності до роботи мікросхеми компанії MicroSemi (Actel) використовуються у військовій і аерокосмічній галузях [36]. Варто відзначити розроблену компанією MicroSemi (Actel) технологію Fusion, що

дозволяє об'єднати логічні блоки FPGA, flash-пам'ять і аналогові пристрої на одній мікросхемі.

Продукція компанії QuickLogic [37], що розпочала свою діяльність в 1988 році, повністю орієнтована на ринок портативних пристроїв. До 2007 року основним її продуктом була серія недорогих одноразово програмованих ПЛІС PolarPro [38] з низьким енергоспоживанням, а потім компанія переорієнтувалася з випуску мікросхем типу FPGA на виробництво CSSP-мікросхем (Customer Specific Standard Products). До основних властивостей CSSP-пристроїв компанії QuickLogic можна віднести: одноразову програмованість, реалізацію процесу програмування самою компанією за участю користувача, наявність в цих пристроях блоків двох типів-програмованих блоків та вбудованих апаратних блоків, що реалізують функції доступу до USB та інші інтерфейси. Основна лінійка продукції QuickLogic застосовується в смартфонах, портативних мультимедійних програвачах, портативних навігаційних пристроях, HDD-накопичувачах, flash-накопичувачах тощо.

Lattice Semiconductor є одним з лідерів в області виробництва мікросхем програмованої логіки FPGA та CPLD. На ринку компанія пропонує цілий спектр ПЛІС різної спрямованості, таких, як CPLD загального призначення, CPLD з низькою потужністю споживання, CPLD з гібридною архітектурою (наприклад серія MachXO [39] - має деякі властивості архітектури типу FPGA, що забезпечує більшу гнучкість при програмуванні), CPLD серії ispXPLD 5000V/B/C, що складається із блоків Multi-Function Block (MFB), кожний з яких може бути запрограмований окремо як обчислювальний блок, блок пам'яті RAM, або як блок, що реалізує буфер типу FIFO, SPLD-мікросхеми, що використовуються для проведення простих операцій або для реалізації міжз’єднань логічних схем на платі. Мікросхеми типу CPLD нової лінійки ICE40 [40] започаткували нові стандарти наднизького енергоспоживання та надмалого розміру. Ці

мікросхеми користуються попитом при виробництві нових смартфонів, цифрових камер, електронних книг та компактних вбудованих систем. Компанія Lattice Semiconductor з'явилася на ринку ПЛІС типу FPGA відносно недавно, але вона пропонує великий асортимент мікросхем даного типу.

Як приклад розвитку сучасних тенденцій та характеристик мікросхем програмованої логіки проведемо огляд сучасної лінійки продукції світового лідера галузі фірми Xilinx [41]. В основі лінійки продукції мікросхем програмованої логіки фірми Xilinx знаходиться Серія 7 [25] ПЛІС з архітектурою FPGA, що включає в себе три нових сімейства. Продукція цих сімейств здатна повністю задовольнити вимогам по створенню сучасних цифрових систем: починаючи від низької ціни, малих габаритів і ефективності використання в серійному виробництві, і закінчуючи надвисокою швидкістю передачі даних, великим обсягом логічних ресурсів і ресурсами цифрової обробки сигналів. До складу Серії 7 входять наступні сімейства (табл.1.1):

- сімейство Artix-7: характеризується низькою вартістю, низьким енергоспоживанням і випускається в компактних корпусах, що орієнтує його для використання в серійних або недорогих пристроях;
- сімейство Kintex-7: має оптимальне співвідношення "ціна/швидкодія" і по даній характеристиці перевершує попередні покоління ПЛІС в 2 рази [42];
- сімейство Virtex-7: кристали мають найвищу в мікросхемах Серії 7 швидкодію та містять в 2 рази більше логічних ресурсів ніж покоління ПЛІС Virtex-6 [25].

Таблиця 1.1 – Порівняння сімейств ПЛІС Серії 7

Макс. значення для сімейства	Artix-7	Kintex-7	Virtex-7
Логічні комірки	360 К	478К	1 995К
Загальна ємність BRAM	19 Мбіт	34 Мбіт	68 Мбіт
Секції DSP	1040	1920	3600

Пікова швидкодія DSP	1248 GMACs	2845 GMACs	5335 GMACs
Гігабитні трансівери	16	32	96
Максимальна швидкість трансивера	6.6 Гбіт/с	12.5 Гбіт/с	28.05 Гбіт/с
Пікова пропускна здатність	211 Гбіт/с	800 Гбіт/с	2784 Гбіт/с

Таблиця 1.1 (продовження)

Підтримка PCI Express	Gen2 x4	Gen2 x8	Gen 3x8
Контролер пам'яті	1066 Мбіт/с	1866 Мбіт/с	1866 Мбіт/с

Кристали Серії 7 [25] виготовляються за новітньою технологією з проектними нормами 28 нм з використанням діелектрика з великим коефіцієнтом діелектричної проникності (HKMG - High-K/MetalGate). Застосування цієї технології дозволяє досягти збільшення загальної пропускної здатності каналів вводу/виводу (до 2.9 Тбіт/с), збільшення максимальної логічної ємності до 2-х млн. логічних комірок, і збільшення швидкодії блоків цифрової обробки сигналів (DSP) до 5.3 TMACS, при цьому енергоспоживання знижене на 50% відносно попередніх поколінь.

До основних особливостей ПЛІС Серії 7:

- швидкодіючі логічні ресурси, що реалізовані на базі 6-входові таблиць перетворення (6-LUT), які також можуть бути конфігуровані як розподілена пам'ять;
- двопортова блокова пам'ять BlockRAM (36 Кбіт) з вбудованою логікою FIFO;
- блоки вводу-виводу, що використовують технологію SelectIO і підтримують високошвидкісні диференціальні сигнальні стандарти, включаючи підтримку інтерфейсу пам'яті DDR3 (до 1866 Мбіт / с);

- високошвидкісні трансівери, що дозволяють реалізувати послідовну передачу даних зі швидкістю від 600 Мбіт/с до 28 Гбіт/с і підтримують спеціальний режим низького енергоспоживання та оптимізовані для міжчипового обміну даними;
- подвійні 12-бітові аналого-цифрові перетворювачі (XADC) загального призначення, із швидкодією 1 млн. вибірок в секунду, які мають вбудовані датчики контролю температури і напруги;
- блоки DSP48 з помножувачем 25x18, 48-бітним акумулятором і предсуматором для високошвидкісної фільтрації, включаючи можливість оптимального побудови фільтрів із симетричними коефіцієнтами;
- блоки управління та синтезу сигналів синхронізації дозволяють забезпечити високу точність сигналів і низький рівень джиттера;
- апаратні блоки PCI Express, що підтримують x8 Gen3 Endpoint і RootPort;
- широкий набір режимів конфігурації, включаючи можливість шифрування конфігураційної послідовності за алгоритмом AES (256 біт) з HMAC/SHA-256 аутентифікацією, і вбудований модуль виявлення і корекції однократних помилок;
- кристали виробляються по проектній нормі 28 нм, з використанням технологій HKMG та HPL (напруга живлення ядра 1.0 В), також доступні кристали з напругою живлення ядра 0,9 В, та зменшеним енергоспоживанням.

Іншим, популярним сімейством ПЛІС архітектури FPGA, що орієнтоване на універсальне застосування, є сімейство Spartan-6 [43]. На його основі можна створювати ефективні та недорогі цифрові пристрої, здатні функціонувати як самостійно, так і в складі мікропроцесорної системи. ПЛІС серії Spartan є розвитком базової серії XC4000 (сімейства XC4000E, XC4000XLi XC4000XLA). Розвиток архітектур ПЛІС відбувався

по шляху реалізації як найбільш швидкодіючих технічних рішень, так і шляхом максимального спрощення і здешевлення мікросхем із збереженням їх основних функцій. Подібне розділення і привело до появи двох сімейств, що розвивалися паралельно: високопродуктивних FPGA серії Virtex і більш дешевої серії Spartan, яка позиціонується як заміна спеціалізованим мікросхемам для пристроїв, що випускаються невеликим тиражем.

Як і ПЛІС попереднього покоління, представленого сімействами Spartan-3, Spartan-3E, Spartan-3A, Spartan-3AN і Spartan-3A DSP [44], кристали з архітектурою FPGA серії Spartan-6 призначені, насамперед, для всебічного використання в складі серійної апаратури різного призначення. Зокрема, ПЛІС даної серії доцільно використовувати для реалізації контролерів високошвидкісних інтерфейсів, високопродуктивних пристроїв цифрової обробки сигналів, вбудованих мікропроцесорних систем, що базуються на 32-розрядних процесорних софт-ядрах сімейства MicroBlaze [45], пристроїв автомобільної електроніки, систем відеоспостереження. Оптимальне співвідношення вартості і функціональних можливостей кристалів даної серії обумовлює їхнє застосування як ефективної заміни ASIC.

Дуже цікавим окремим напрямком розвитку технологій ПЛІС, який останнім часом набуває поширення, є мікросхеми класу EPP (Extensible Processing Platform або процесорна платформа з можливістю розширення). Одними з перших в цьому класі фірмою Xilinx було створено сімейство Zynq-7000, що відносяться до групи пристроїв „All programmable SoC”. Така мікросхема включає в себе популярне сьогодні процесорне ядро ARM CortexA9 [46] та матрицю програмованих логічних комірок, аналогічну до FPGA, сімейств Artix або Kintex. Відмінністю від класичних мікросхем з архітектурою FPGA є те, що процесорна система виконана на кристалі разом з пам'яттю і

периферійними пристроями та повністю готова до роботи. В той же час, потенціал програмованих ресурсів за продуктивністю в десятки або сотні разів вище ніж у процесорної підсистеми. Це дозволяє на базі мікросхем цього класу створювати системи з високою продуктивністю в яких обробка даних виконується паралельно за рахунок поєднання обчислювальних потужностей програмованої частини FPGA та процесорної платформи ARM. Таким чином, новітній клас мікросхем Zynq-7000 надає нові можливості для створення вбудованих систем, що дає розробникам гнучку платформу для нових недорогих [31] рішень замість традиційних ASIC і ASSP.

З огляду на питання застосування ПЛІС-технологій як основи для обчислювальних засобів, що реалізують пошукові операції, можна окреслити наступні переваги цього підходу:

1. Продуктивність ПЛІС-системи, що реалізована як самостійний модуль, не знижується внаслідок затримок, які виникають при організації міжпроцесорного обміну, обміну даними із загальною пам’яттю та вводу/виводу.

2. Системи на базі ПЛІС характеризуються низькою затримкою реакції та детермінованістю відгуку [4].

3. Можливість швидкого переналаштування структури обчислювача в залежності від параметрів пошуку завдяки такій властивості ПЛІС, як реконфігуровність [12-18].

1.2. Основні особливості архітектура ПЛІС типу FPGA

Основною особливістю FPGA є наявність матриці логічних елементів, що мають від двох до шести входів, тригерів, відрізків ліній зв'язку, що з'єднуються перемичками з польових транзисторів. Field Programmable Gate Array програмуються зміною напруженості електричного поля (field) в затворах цих транзисторів. Затвори всіх таких програмовних польових

транзисторів підключені до виходів тригерів багаторозрядного регістра зсуву, в який записуються відповідні двійкові значення для програмування ПЛІС. Деякі з ділянок цього регістра можуть також виконувати роль оперативних (ОЗП) або постійних запам'ятовуючих пристроїв (ПЗП). Програма (прошивка) звичайно зберігається в ПЗП, що використовується разом із ПЛІС. Після подачі живлення або за сигналом скидання вона автоматично переписується в програмуючий регістр зсуву ПЛІС – так конфігурується ПЛІС [47].

Розглянемо типову архітектуру FPGA на прикладі кристалів серії Virtex 4 фірми Xilinx. На кристалі (рис. 1.1) розташовано основні елементи ПЛІС: матриця конфігуровних логічних блоків (англ. Configurable Logic Block, CLB); матриця програмовних електронних ключів міжз'єднань (польових транзисторів із плаваючим затвором, що розташовані на перетині вертикальних та горизонтальних ліній) для утворення ліній електричного зв'язку між внутрішніми блоками FPGA; блоки ОЗП (англ. BlockRAM), які розміщені уздовж кристалу; блоки DSP-48 (англ.Digital Signal Processing) з помножувачем 25x18, 48-бітним акумулятором і передсуматором для високошвидкісної фільтрації; блоки вводу-виводу (англ. Input Output Block, IOB) [48].

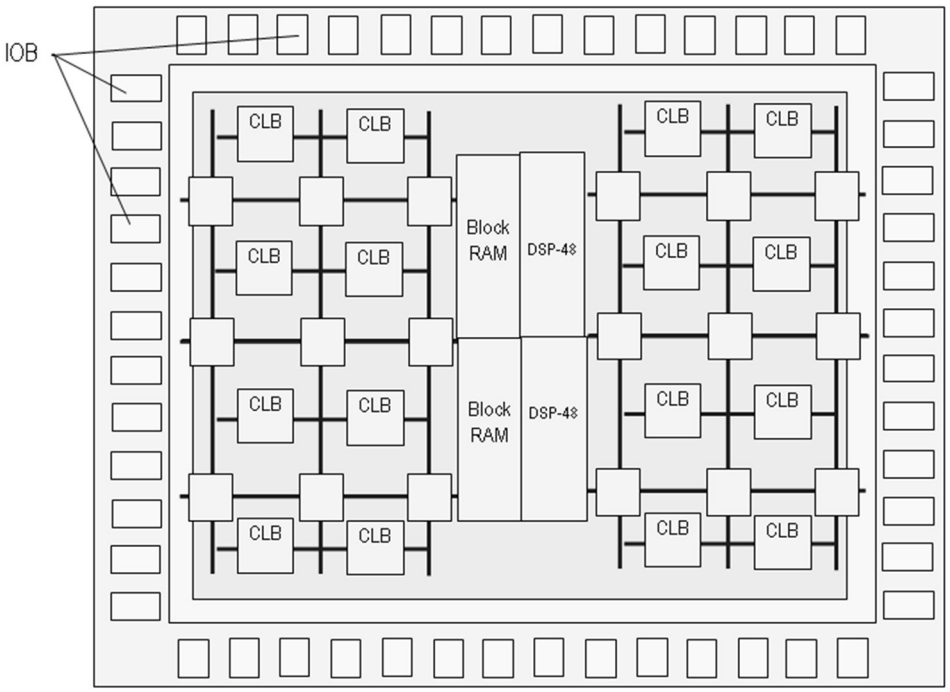


Рисунок 1.1 – Структура ПЛІС

Основним логічним елементом в ПЛІС є логічна таблиця (англ. look-up table, LUT), у вигляді однобітового оперативного запамятовувального пристрою на 16 комірок. Тригери LUT входять до складу програмовного регістру зсуву, і їх початковий стан заповнюється під час конфігурування ПЛІС. В сучасних ПЛІС використовуються LUT підвищеної ємності, які реалізують логічні функції від 6 аргументів. Крім того, інколи окремі частини пам'яті блочного ОЗП (BlockRAM) конфігуруються як багатоаргументна логічна функція. У ПЛІС також використовують програмовні синхронні D-тригери. Завдяки їх програмному конфігуруванню можна задати такі режими роботи: тригер з початковим скиданням (R) або з початковою установкою (S); тригер із записом за фронтом або спадом синхросигналу; тригер із сигналом дозволу запису (E) або без нього. Після закінчення конфігурування ПЛІС видає сигнал загального скидання (англ.global set-reset, GSR), який встановлює усі тригери в 0 або 1. Інші типи тригерів, таких як J-K-, R-S-, T-тригери (звичайно, якщо вони потрібні для проектованого пристрою), реалізуються на основі D-тригерів, до входів яких підключено LUT, яка відповідає необхідній функції збудження [47].

CLB із двох тригерів та двох LUT називають сегментом CLB (англ. CLB slice). При збільшенні кількості входів LUT можливості CLB значно розширюються. Так, із 6-входовими LUT на CLB slice можна реалізувати тривходовий суматор або суматор із мультиплексором на вході. В ПЛІС останніх поколінь з'явилися ресурси для реалізації багатовходових мультиплексорів. Так, 6-входова LUT реалізує 4-входовий мультиплексор замість 2-входового мультиплексора, що було можливим у 4-входовій LUT. Додаткові мультиплексори у складі конфігуровних логічних блоків забезпечують побудову деревоподібних мультиплексорів із 8, 16 і більше входами. Такі мультиплексори необхідні для пересилки операндів в арифметико-логічний пристрій від різних джерел, а також для реалізації внутрішніх загальних шин обчислювальних засобів.

Для створення швидкодіючого однорозрядного ОЗП використовують ресурси LUT ПЛІС. Пари сусідніх LUT можна об'єднати в двопортовий ОЗП. При цьому запис виконується за однією адресою в обидві LUT, а читання – за двома різними адресами. У сучасних ПЛІС ємність такого ОЗП може досягати 64 біт. Для нарощування ємності пам'яті виходи декількох CLB з модулями ОЗП об’єднують через мультиплексори. При цьому необхідні додаткові апаратні витрати для побудови пристрою дешифрування адреси, яка передає сигнали LUT на входи мультиплексора. Таким чином, оперативна пам’ять виявляється розподіленою по площині кристалу і тому має назву розподілений ОЗП (англ. Distributed RAM).

Для зберігання масивів даних та для їх буферизації в ПЛІС використовують від десятків до тисяч окремих блоків ОЗП (англ. Block RAM). Кожний такий блок в ПЛІС фірми Xilinx по суті є двопортовою пам'ятю ємністю 36 Кбіт. Розрядність даних може бути встановлена від 1 до 36 розрядів.

Обчислювальні засоби на ПЛІС знаходять широке застосування завдяки тому, що їх можна легко долучати до більшості проектів цифрових

засобів та, крім того, ними досить просто можна заміняти пристрої, які реалізовані на застарілій елементній базі. Таке долучення реалізується за допомогою великої кількості блоків вводу-виводу (ІОВ), які налаштовують під різні стандарти електричного з'єднання входів мікросхем.

Для діагностики, налагодження та, конфігурування, ПЛІС може бути переключена в так званий режим граничного сканування (Boundary Scan). У цьому режимі всі ІОВ з'єднуються в ланцюжок одного довгого регістру зсуву. Керуючи цим регістром зсуву через інтерфейс JTAG можна зчитувати стан виводів, подавати тестові сигнали, конфігурувати ПЛІС.

Для програмування та конфігурування ПЛІС можуть використовуватись програматори, зовнішні ПЗП та завантажувальні кабелі. Завантажувальні кабелі можуть використовуватись при програмуванні мікросхеми, яка вже встановлена на друковану плату. Така технологія носить назву внутрішньосхемного програмування (англ. In-System programming – ISP).

При використанні зовнішніх ПЗП конфігураційна інформація зберігається в спеціалізованих мікросхемах і при включенні живлення завантажується до ПЛІС. Слід відмітити, що конфігураційні ПЗП відрізняються від звичайних ПЗП тим, що крім збереження інформації вони ще повинні працювати з мікросхемою ПЛІС за стандартним протоколом конфігурування. Протокол залежить від режиму конфігурування. Для мікросхем ПЛІС можливі такі режими програмування:

- пасивний послідовний (англ. Passive Serial);
- асинхронний пасивний послідовний (англ. Passive Parallel Asynchronous);
- пасивний паралельний (англ. Passive Parallel);

– швидкий пасивний паралельний (англ. Fast Passive Parallel);
– програмування через JTAG-інтерфейс;
– активний послідовний (англ. Active Serial).

Наприклад, для конфігурування ПЛІС серії Spartan передбачено спеціальні три входи задання одного із режимів, вивід сигналу синхросерії програмування CCLK, вхід послідовності конфігурації PROGRAM, вихід прапора закінчення конфігурування DONE і виводи порту JTAG. Залежно від встановленого режиму можна завантажувати прошивку ПЛІС одним із трьох способів: через однорозрядний вхід PROGRAM, порт JTAG або 8-розрядну шину D із використанням для управління виводів WRITE та BUSY.

Конфігурування через однорозрядний вхід триває не більше 10 секунд. Це стандартний спосіб конфігурування і для нього не потрібно додаткового обладнання, крім ПЗП прошивки з однорозрядним виходом.

1.3. Засоби автоматизації проектування комп'ютерних засобів на ПЛІС

В сучасних умовах, коли використання ПЛІС в більшості випадків є доцільним для розробки ефективних цифрових обчислювальних пристроїв та враховуючи, що електронні пристрої на їх базі - один із секторів мікроелектроніки який найбільш активно розвивається [49], варто звернути увагу на можливості та особливості засобів, що орієнтовані на вирішення завдання логічного синтезу для мікросхем програмованої логіки. В основі сучасного процесу створення цифрових пристроїв на відміну від попередніх років, коли базовим етапом проектування було створення принципової електричної схеми, знаходиться технологія проектування програмних моделей обчислювальних засобів за допомогою опису апаратних засобів на одній з мов RTL-рівня (рівня міжрегістрових передач) [50], таких як VHDL [51] та Verilog [52], а сам процес проектування схожий на процес програмування – в ролі компілятора виступають системи синтезу.

Для розробки та відлагодження моделей пристроїв використовують спеціальні інтегровані середовища із вбудованими засобами моделювання, відлагодження, компіляції, симуляції, аналізу продуктивності, інтеграції та іншими функціями. Серед них - ModelSim® від Menthor Graphics [53], Active-HDL від Aldec [54], Altium Designer від Altium [55], Socrates від Duolog technologies [56].

На етапі логічного синтезу обчислювальних засобів також використовують програмні засоби логічного синтезу ПЛІС, що надаються фірмами-виробниками цих мікросхем, які представлені у вигляді спеціальних інтегрованих середовищ. Такі засоби розробки надаються виробниками ПЛІС: Altera, Xilinx, Microsemi (Actel), Lattice Semiconductor, QuickLogic, Cypress та ін. Кожна з цих компаній надає користувачам засоби логічного синтезу своїх виробів, наприклад, фірма Altera – Quartus II, Max+Plus II [57], фірма Xilinx - ISE [58], Vivado [59]. Для конфігурування ПЛІС використовують виготовлені фірмами-виробниками ПЛІС спеціальні засоби, що складаються із друкованої плати, на яку поміщено кристал ПЛІС, та допоміжних для засобів програмування ПЛІС. Необхідне програмне забезпечення постачають окремо чи в комплекті з апаратними засобами, а інколи його вбудовують у засоби логічного синтезу ІІЛІС.

Як приклад, розглянемо характеристики та особливості нового покоління САПР, що призначене, насамперед, для роботи з FPGA великого об’єму. Лідер ринку програмованої логіки - компанія Xilinx представила комплект для розробки цифрових пристроїв Vivado [60]. Це середовище системного проектування, яке дозволяє зменшити час створення пристроїв на базі мікросхем класу „All Programmable devices” [24]. Новий набір програмного інструментарію, що включений до пакету, не тільки збільшує ефективність розробки пристроїв на базі мікросхем програмованої логіки, але і прискорює інтеграцію програмованої системи, що може використовувати технологію тривимірних з'єднань багаторівневих

кристалів кремнію та містить процесорну систему, підсистеми змішаного сигналу, аналогові блоки та IP-ядра [22]. Засоби середовища Vivado підтримують нові мікросхеми ПЛІС Серії 7 (Virtex-7, Kintex, Artix), Zynq та орієнтовані на ПЛІС, що містять 50 тис. логічних комірок та більше (даний комплект розробки дозволяє підвищити продуктивність розробників в 4 рази порівнянно з конкуруючими системами [61]), інакше ефект від використання САПР Vivado не буде суттєвим. Необхідність в новому інструменті розробки проектів на базі ПЛІС пояснюється підвищенням складності проектів, та переходом до маршруту проектування, що використовує численні IP-компоненти. САПР Vivado, також, забезпечує можливість розширеного аналізу проекту та зручної інтеграції компонент. Для цього Vivado має набір засобів для візуалізації та аналізу зв'язків між компонентами проекту, управління часовими обмеженнями, інтеграції в проект модулів, що створені сторонніми розробниками. У Vivado існує підтримка мов високого рівня (HLS, High-Level Synthesis). Таким чином, розробку можна вести не тільки з застосуванням мов опису апаратури (VHDL, Verilog), але і на C/C++/SystemC [62-65].

Отже, комплект розробки Vivado представляє собою високоінтегроване середовище проектування (IDE) з абсолютно новими інструментами проектування системи, побудованими на основі масштабованої моделі, що спільно використовується та загального середовища відлагодження. У наборі інструментів Vivado поєднуються всі види технологій програмування та реалізована підтримка проектів, що містять до 100 млн. еквівалентних вентилів ASIC. З метою розширення можливостей інтегрування системи в набір проектування Vivado включені: інструменти розробки системного рівня для швидкого синтезу і верифікації IP-алгоритмів на базі C/C++/SystemC; стандартизовані IP-блоки для розробки корпусних конструктивів; стандартизовані системні IP-блоки і блоки верифікації та моделювання з високою продуктивністю.

Для покращення можливостей реалізації проекту в склад інструментів системи Vivado входять ієрархічний редактор пристроїв та засіб розробки компонування кристала, інструмент логічного синтезу з більш високою продуктивністю з підтримкою SystemVerilog, продуктивна підсистема розміщення компонент і розводки з функціями оптимізації по затримках і довжині міжз'єднань. Крім того, маршрут проектування допускає технологію зміни порядку проектування ECO (Engineering Change Orders), що дозволяє модифікувати лише невеликі частини проекту, зберігаючи високий темп проектування. Використання нової масштабованої моделі даних дозволяє виконувати оцінку енергоспоживання, часових параметрів на кожному етапі проектування, що забезпечує швидкий аналіз і оптимізацію конфігурації розроблювальної обчислювальної системи.

1.4. Огляд та класифікація загроз та атак на блоки інтулектуальної власності в проектах на ПЛІС

Важливість забезпечення інформаційної стійкості комп’ютерних пристроїв, що реалізовані на ПЛІС в розрізі протидії негативним явищам, таким як несанкціоновані доступ до інтелектуальної власності або її крадіжка підтверджується наявністю загроз та слабких сторін у таких апаратних засобів [66].

При проектуванні сучасних спеціалізованих моніторингових засобів на ПЛІС використовуються готові блоки (модулі) інтелектуальної власності - IP (intellectual property), IP-блоки або IP-core що втілені у функціональних можливостях пристроїв та які реалізують досить складні алгоритми перетворення і обробки даних. IP - це складний, вже протестований, веріфікований і оптимізований функціональний модуль, який може бути багато разів використаний як компонент при проектуванні більш складних цифрових виробів з метою скорочення часу їх розробки. Як показала практика, використання IP дозволяє скоротити час до моменту появи

цифрового виробу на ринку, спрощує процес проектування шляхом створення інтерфейсу для зв'язку системи з IP, мінімізує ризики проектування завдяки включенню вже відлагоджених модулів, зменшує час верифікації всієї системи. Сучасний ринок IP постійно розвивається та розширюється [22]. Але існує і зворотній бік - те, що створюється місяцями або рокам може буде вкрадено за секунди. Викрадання блоків інтелектуальної власності (IP) можливе при клонуванні (копіюванні) пристрою або при зворотному проектуванні (інжинірінгу) [67]. Якщо в першому випадку створюється пристрій максимально наближений до оригіналу, що не завжди можна вважати крадіжкою, то в другому випадку зловмисник дістається до самої суті інтелектуального продукту – алгоритмів функціонування пристрою та побудови нової специфікації пристрою для подальшої маніпуляції цією інформацією.

Серед інших загроз для IP можна виділити такі як надлишковий випуск продукції, коли компанії, на потужностях яких випускають мікроелектронні вироби (OEM) продукують надлишкові партії без дозволу компанії-замовника; злам в результаті якого, аналогічно до зворотнього проектування, отримується інформація про виріб та здійснюється подальше маніпулювання робочими станами пристрою на основі ПЛІС, наприклад, його вимикання. В більшій мірі слабкі сторони проектів сприяють реалізації загроз. Серед таких сторін виділяють наявність халатного ставлення розробників до наявності загроз; невиконання користувачами IP інструкцій із заходів безпеки або повне їхнє ігнорування; наявність backdoor (дірок у захисті) які залишають розробники пристрою на етапі відлагодження або тестування; помилки в сфері безпеки проекту; помилки в роботі пристроїв на ПЛІС– одиничні збої SEU (англ. Single-Event Upset), що виникають внаслідок дії на чутливі структури його пам’яті часток із високою енергією (англ. Single-Event Effect, SEE) в результаті чого змінюється стан одного або декількох бітів даних в пам'яті [68-70].

Провідні виробники мікросхем ПЛІС мають засоби для протидії таким загрозам та усунення впливу факторів, що є причиною тієї чи іншої слабкої сторони. Але не зважаючи на це, також, існують атаки на ПЛІС-реалізації спеціалізованих комп'ютерних засобів. До таких атак відносять дії по декодуванню конфігураційної послідовності (bitstream) з метою відновлення netlist як один із етапів зворотного інжинірингу. Процес генерування bitstream вважається закритим - у кожного постачальника ПЛІС є своє фірмове рішення і вони не надають засобів для зворотнього перетворення bitstream у netlist. Неможливо чітко співставити конфігураційну послідовність або її частину деякому списку з'днань netlist - для реалізації цього можуть знадобитись значні ресурси. Тому формування конфігураційної послідовності розглядається як одна із ланок захисту IP оскільки у вигляді bitstream проект стає неочевидним, заплутаним та незрозумілим для аналізу.

Особливим випадком зламу є підміна, коли зловмисник заміняє (змішує) всю або частину bitstream FPGA-проекту з власною частиною конфігураційної послідовності та видає за власний проект.

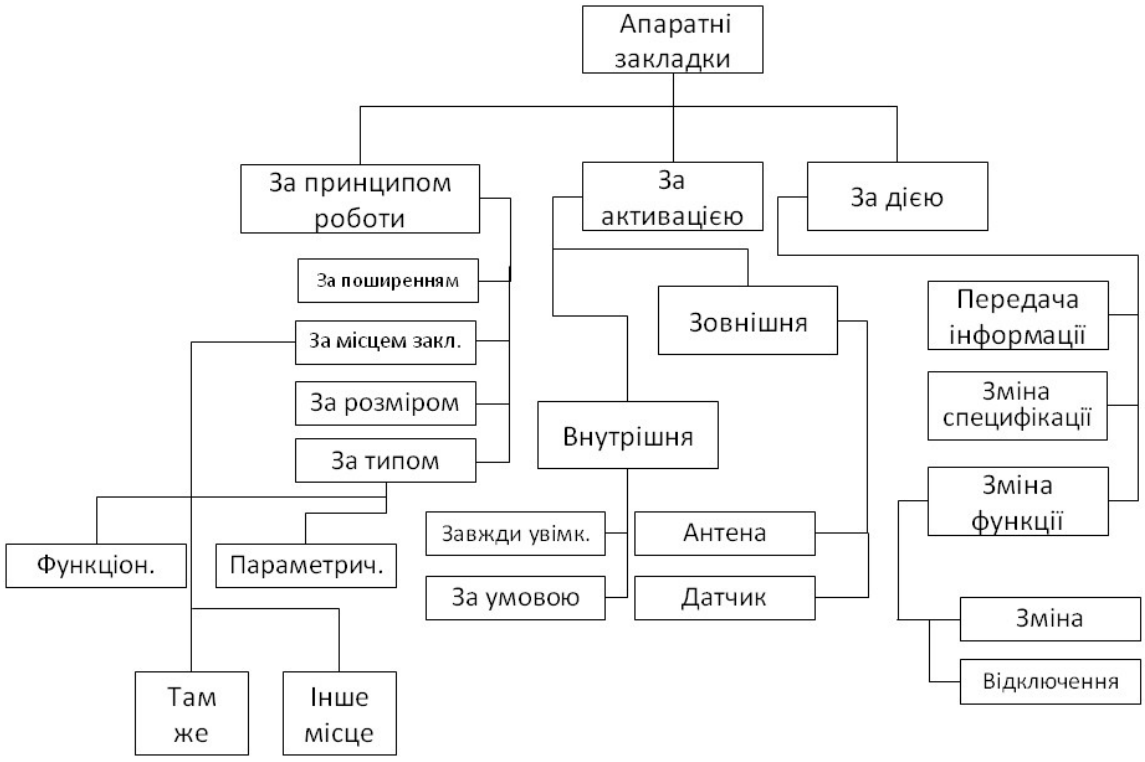


Рисунок 1.1 – Класифікація апаратних закладок

Апаратним троянцем (англ. Hardware Trojan, trojan) або апаратною закладкою будемо називати частину мікроелектронного виробу яка нелегально та таємно вбудовується в пристрій та здатна втрутитися в роботу обчислювальної системи (рис1.1). Їх можна вважати шкідливою зміною апаратної специфікації чи реалізації, що змінює функціональність IP. Апаратні троянці є більш небезпечними ніж програмні троянці через те, що вони знаходяться на найнижчому рівні обробки інформації і продовжують нести загрозу системі до тих пік поки використовується інфіковане апаратне забезпечення.

Апаратні закладки можуть бути фізично розташовані в різних місцях мікрочіпа. У деяких випадках зловмиснику доводиться серйозно змінити проект, і чим непомітніше він зробить ці зміни, тим складніше буде знайти закладку. Також існують закладки, які встановлюються окремо від пристрою. Характер змін, що вносить зловмисник теж має велике значення для подальшого виявлення та знешкодження закладки. Сюди можна віднести кількість змінених, доданих або видалених елементів проекту. Також, апаратні закладки можуть впливати на функціональність обчислювального пристрою шляхом додавання нових функцій, а, також, шляхом впливу на правильне функціонування вже існуючих компонент. Активація троянців може бути викликана зовнішніми так і внутрішніми факторами. У першому випадку для запуску закладки використовується зовнішній сигнал, який приймається антеною або датчиком, якщо такі наявні в системі. Сигналом від датчика може бути результат будь-якого вимірювання: температури, висоти, тиску, напруги тощо. У випадку реализації активізації троянця внутрішнім способом не потрібна взаємодія із зовнішнім засобами. В цьому випадку апаратна закладка або функціонує весь час або запускається за певної умови, що закладена зловмисниками. За характером негативного впливу на комп’ютерні засоби або наслідків, що

вони спричиняють, апартні закладки можна поділити на такі, що здійснюють передачу інформації, порушують роботу всього пристрою або тільки певної функції. При чому зміна функціональності здійснюється шляхом її зміни або повного відключення.

Результатом роботи апаратної закладки може бути як повне виведення системи з ладу, так і порушення її нормального функціонування, наприклад несанкціонований доступ до інформації, її зміна або блокування. Атаки із застосування апаратних троянців або апаратних закладок проводяться для того щоб отримати доступ до інформації, що зберігається в ПЛІС або, навіть, викрасти IP. Засоби такого типу зловмисники вбудовують в програмне забезпечення САПР або в саму логіку проекту, тобто в IP. В результаті цього проект стає вразливим до зламу вже в процесі розробки.

Так зване зворотне зчитування bitstream з ПЛІС можна віднести до загроз IP оскільки за допомогою цих даних можна отримати інформацію про стан елементів пам'яті користувача, стан внутрішніх регістрів системи.

Зворотне зчитування може потенційно використовуватися для відтворення конфігураційної послідовності проекту.

Атаки за бічним каналам (Side-channel attack) [71] зловмисники використовують операційні характеристики системи та інформацію про фізичні процеси у пристрої, наприклад атаки по часу, по енергоспоживанню, по електромагнітному випромінюванню, за помилками обчислень тощо. За допомогою таких атак зловмисники отримують ключі шифрування та інформацію про проект.

Атака за допомогою внесення несправностей - метод, який використовується в тестуванні апаратних засобів. Передбачає штучне внесення різного роду несправностей для тестування відмовостійкості і, зокрема, виникнення ситуацій обробки виняткових ситуацій шляхом примусу перейти пристрій на ПЛІС в режим тестування, відлагодження, неправильного стану або виводу закритої інформації. Зловмисник створює

аварійні ситуації в роботі пристрою змінюючи конструкцію, умови навколишнього середовища, напругу або температуру. За допомогою таких дій в ПЛІС-пристроях можуть змінюватися біти у конфігураційній послідовності, щоб впливати на функціональність. Проте, прийоми, що застосовуються при проектуванні сучасних апаратних засобів, такі як визначення всіх станів та повний аналіз збоїв, роблять цей тип атак на FPGA складним для реалізації.

Таким чином загрози та атаки, що супроводжують процес реалізації обчислювальних засобів на ПЛІС, і сприяють виникненню, в першу чергу, проблем із використанням блоків інтелектуальної власності, захист яких від несанкціонованого ознайомлення, використання, підробки, модифікації тощо є актуальною задачею забезпечення інформаційної стійкості цих засобів.

1.5. Висновки до розділу 1

В результаті проведеного аналізу задач, сучасних форм цифрових пристроїв, лінійок продукції фірм-виробників напівфабрикатів ПЛІС та згідно наведених статистичних даних показано перспективність та ефективність використання мікросхем програмовної логіки при реалізації спеціалізованих моніторингових комп'ютерних засобів.

Також, в даному розділі показано, що проблема забезпечення необхідного рівня достовірного функціонування постає гостріше, оскільки комп'ютерні засоби піддаються впливу негативних факторів та явищ як на етапі проектування та реалізації, так і на етапі функціонування. Проведено огляд та класифікацію атак та загроз на блоки інтелектуальної власності.

Таким чином, огляд проблемних питань використання комп'ютерних моніторингових засобів на ПЛІС, їхніх властивостей та особливостей показав актуальність проблеми забезпечення протидії атакам та загрозам через розробку нових концепцій реалізації такого захисту.

РОЗДІЛ 2

СУЧАСНІ РЕАЛІЗАЦІЇ ЗАХИСТУ БЛОКІВ ІНТЕЛЕКТУАЛЬНОЇ ВЛАСНОСТІ СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ ЗАСОБАХ НА ПЛІС

2.1. Ефективність реалізацій захисту IP в проектах комп'ютерних засобів на ПЛІС

Існують класичні підходи до реалізації захисту IP [72-73]. Сюди відносять захист за допомогою патентів, копірайту, юридичних механізмів та впровадження комерційної таємниці й організаційно-правових заходів на підприємстві. Наприклад, в деяких країнах зворотнє проектування допускається тільки із метою навчальних або дослідницьких цілей. Інший класичний підхід до захисту використовує шифрування проекту, коли разом із IP постачальник передає клієнту ключ для дешифрування, але, знову ж таки, має укладатися угода про використання IP та ключа.

Ефективним із точки зору безпеки IP є розподіл конфіденційних даних по проекту між різними компаніями, що беруть участь у виробництві

комп’ютерних засобів на ПЛІС так, щоб запобігти можливості отримати доступ до всієї інформації. За своєю сутністю ПЛІС є безпечною базою для реалізації обчислювальних засобів оскільки при виготовленні кінцевий продукт відокремлено від проектувальника виробником устаткування (OEM).

Провідні виробники ПЛІС [74] пропонують широкий спектр рішень для забезпечення захисту проектів (IP) - починаючи від впровадження ідентифікатора пристрою (DNA) та шифрування конфігураційної послідовності до використання механізму перевірки цілісності інформації (HMAC) [75], для аутентифікація бітового потоку, а також використання спеціалізованих засобів для безпеки.

Шифрування bitstream в процесі реалізації комп’ютерних засобів на ПЛІС служить як для запобігання клонування пристрою, так і для захисту конфіденційних конфігураційних даних розробки. Кожен напівфабрикат ПЛІС-пристрою має в своєму складі спеціальний блок для розшифровки bitstream для підтримки зашифрованих стандартним шифруванням AES бітових потоків [76].

Спрощено, опис прикладу реалізації [77] шифрування bitstream проекту можна наступним чином. Система шифрування бітових потоків складається з двох частин: програмного забезпечення шифрування бітових потоків та системи розшифрування бітового потоку на основі мікросхеми пам'яті для зберігання 256-бітового ключа шифрування. Використовуючи програмне забезпечення для проектування, розробник проекту створює як ключ для шифрування, так і зашифрований бітовий потік. Далі ключ шифрування зберігається в спеціальній енергонезалежну RAM ПЛІС або у регістрах самоналаштовуючих чіпів (чіпів eFUSE) [78]. Ключ шифрування можна переслати в пристрій лише через порт JTAG. Під час конфігурування ПЛІС виконує зворотну операцію, розшифровуючи вхідний bitstream за допомогою блоку дешифрування ПЛІС за алгоритмом AES. Розташований на

кристалі ПЛІС блок AES дешифрування не доступний для розробника проекту і не може використовуватися для розшифрування ніяких інших даних, крім конфігураційної послідовності. Як додатковий рівень безпеки, використовується заборона завантаження в ПЛІС зашифрованого bitstream. Розшифрований потік біт не може бути завантажений в FPGA, якщо конфігураційна пам’ять повністю не очищена. Також, неможна завантажувати в ПЛІС незашифровані проекти якщо ПЛІС завантажена зашифрованим bitstream. Це допомагає протидіяти атакам зворотного інжинірингу.

Для захисту від атак підміни та троянців свою ефективність демонструють засоби, що використовують криптографічно стійку аутентифікація [79]. Для передачі зашифрованого алгоритмом AES bitstream використовує вбудований в ПЛІС блок для хешування потоку за способом перевірки автентичності повідомлень HMAC [80]. Використовуючи механізм перевірки цілісності інформації HMAC, засоби проектування створюють код автентифікації повідомлень (англ. Message Authentication Code, MAC), використовуючи секретний ключ і саме повідомлення [81]. Приймаюча сторона (в даному контексті в програмовному середовищі, апаратна сторона) обчислюється цей хеш-код (MAC) для отриманого повідомлення за допомогою того ж самого ключа та порівнюються результати. Обидві компоненти генерують 256-бітний MAC на основі ключа та захищеного алгоритму хешу SHA256. Якщо ці два значення співпадають, то повідомлення вважається перевіреним. Так без знання ключів для AES та HMAC bitstream IP не можна завантажити, модифікувати або клонувати. Якщо за допомогою алгоритму AES забезпечується захист вмісту IP від копіювання або зворотного проектування, то використання механізму хешування HMAC гарантується, що bitstream конфігурації пристрою, які завантажується в ПЛІС не було змінено. Таким чином виявляється будь-яка

зміна в конфігураційному потоці, включаючи зміну значення хоча б одного біту.

Для майже повного виключення загроз для IP часто застосовують додаткові методи для забезпечення безпеки які використовуються в поєднанні з методами безпеки, що писані вище. Так можуть використовуватися стійкі до одиночних збоїв (SEU) схеми відключення зворотного з'єднання із потрійним резервуванням, яка блокує зчитування конфігураційної пам'яті через інтерфейс обміну, і що направлено проти прийомів зворотної інженерії.

2.2. Підходи до організації захисту від одиночних збоїв

Останні дослідження [68] вказують на недосконаліть у відношенні надійності сучасних напівпровідникових кристалів, що виготовлені за зниженими технологічними проектними нормами. Серед причин цього, що вже вказувалось, виділяють таке явище, як SEE – разовий вплив на роботу мікросхеми часток із високою енергією (важкого іону або протону) [69-70]. Очевидно, що з такими явищами не можна не рахуватися при застосуванні виробів мікроелектронної промисловості в космічній галузі, де вони найбільше розповсюджені. Наслідки, що спричинені SEE, можуть мати різний характер: деструктивні випадки (SEB або SEL) – hard errors; недеструктивні (SEU, SET, SEFI) – soft errors. Щодо деструктивних випадків, то проблемами відмов та ліквідації їх наслідків займається теорія надійності. До таких випадків відносять SEB - (англ. Single-Event Burnout) – ефект разового вигорання та SEL - (англ. Single-Event lutchup) – ефект разової засувки.

SEU може впливати на комірки пам'яті та призводить за собою зміну значення одного або декількох біт і тут важливо, де ця пам'ять розташовується. Спотворені біти можуть належати конфігураційній пам'яті, яка містить проект. Ця пам'ять найбільша за об'ємом та фізично розподілена

по частинам пристрою, але лише частина бітів має важливе значення для правильної роботи будь-якого конкретного проекту, що завантажено в пристрій. Інші елементи пам’яті високої ємності, які використовуються для зберігання стану проекту називають блочною пам’яттю. Ця пам’ять є другою за ємністю. Елементи блочної пам’яті об’єднані в групи та розташовані по всій ПЛІС. До розподіленої пам'яті належать запам’ятовуючі елементи для зберігання стану проекту на ПЛІС. Цей тип пам'яті реалізовано на базі матриці конфігуровних логічних блоків (CLB) та розподілено по всьому програмовному кристалу. Розподілена пам'ять представляє собою третю за ємністю пам’ять. Останній вид пам’яті базується на тригерах, має невелику ємність і використовуються для зберігання стану проекту. Цей тип пам'яті присутній у всіх CLB і розташований по всій мікросхемі ПЛІС. Тригерна пам’ять має найвищу швидкодію та є найдефіцитнішим програмовним ресурсом ПЛІС.

Для боротьбі із одиночними збоями (SEU), а також атаками за бічним каналом та зламом застосовують спеціальні засоби в ПЛІС що реалізують постійне зчитування у фоновому режимі конфігураційної послідовності проекту. Разом із цим використовується механізм виявлення та корекції помилок, що відомий як error-correcting code (ECC) [82]. Даний підхід спрощує виявлення змін у конфігураційній пам’яті, що можуть виникати або в результаті SEU, або через атаку на проект. Розглянемо особливості функціонування такого механізму. Ці засоби ПЛІС для боротьби із одиночними збоями не запобігають появі помилок, але дозволяють або пом’якшувати наслідки або виправляти помилки.

Оцінимо характерні особливості цих засобів. Реалізація пом’якшення впливів одиночних збоїв у розглянутих типах пам’яті ПЛІС може бути створена шляхом впровадження в проект пристрою блоків, що виявляють та корегують помилки або застосовують надмірність. Помилки, які виникли в результаті SEU в логічних ресурсах ПЛІС що не зайняті проектом

ігноруються. Також використовується класифікація помилок на «важливі» та «несуттєві». Без такої класифікації для конфігураційної пам’яті необхідно враховувати та корегувати всі помилки.

До особливостей реалізацій відомих рішень [83] можна віднести:

- невеликий час виявлення збоїв – близько 20-30 мс;
- використання вбудованих в програмовний кристал апаратних примітивів для виявлення та корекції помилок;
- корекція методом відновлення із використанням тільки алгоритму ECC або разом із алгоритмом CRC [84];
- корекція методом повної заміни шляхом повторного завантаження даних;
- визначення міри впливовості помилки на функціональність проекту;
- збільшення часу безперервної роботи пристрою на ПЛІС за рахунок уникання або нівелювання процедур відновлення роботоздатності пристрою для „несуттєвих” помилок.

Останні моделі пристроїв програмовної логіки мають можливість знищувати ключ шифрування із відповідних комірок пам’яті або вміст конфігураційної та тригерної пам’яті ПЛІС за спеціальним сигналом. Такий маханізм може вмикатися у відповідь на дії зловмисника.

Для запобігання клонування пристрою, ПЛІС останніх поколінь містять вбудований унікальний ідентифікатор пристрою. Цей унікальний ідентифікатор (за аналогією із серійним номером) зберігається у енергонезалежній пам’яті і заносить туди на останньому етапі програмування кристала. В проекті пристрою користувач може використати цей унікальний ідентифікатор для реалізації свого механізму захисту IP від зламу.

2.3. Підходи та існуючі реалізації ідентифікації та захисту блоків інтелектуальної власності від апаратних троянців

Виявлення та нівелювання впливу апаратних атак на блоки інтелектуальної власності із кожним днем стає більш актуальним, оскільки, як було сказано в Розділі 1, використання численних IP-блоків від різних постачальників при реалізації сучасних комп'ютерних засобів сприяє зниженню захищеності проектів.

IP-блок представляє собою оптимізований віртуальний модуль, який дозволяє зменшити виробничі витрати, але, із іншого боку, відкриває шлях до заволодіння зловмисниками інтелектуальним продуктом у вигляді тих же самих IP. Така ситуація склалася тому що більшість компаній-виробників тепер не мають власних виробничих потужностей і вдаються до виробництва своєї електронної продукції на потужностях сторонніх компаній. До того ж, на даний час проекти є настільки складними, що фірми-виробники схиляються до використання IP-блоків від третіх сторін у власних проектах. Однак, використання IP-блоків від сторонніх постачальників (проектувальників) викликає багато питань щодо захищеності проекту. Деталі IP-блоків є прихованими та процес виробництва є замаскованим від замовника з метою захисту самих IP-блоків. Проектувальники мають прийняти на віру факт того, що ніякі шкідливі електронні схеми не вбудовано в їх рішення іншою (третьою) стороною – постачальниками або власниками IP-блоків.

Особливу увагу, заслуговують атаки типу троянський кінь (апаратні закладки) на ПЛІС-проекти. В загальному випадку, виявлення таких закладок необхідно порівняти зразковий модуль IP із тим, що надійшов від постачальника. Але це неможливо, оскільки сама суть IP передбачає, що цей віртуальний модуль є чорною скринькою без усіляких зразків і його використання відбувається на свій страх і ризик. З поширенням використання ПЛІС при реалізації комп'ютерних засобів, що

застосовуються, в тому числі і для моніторингу об’єктів критичного призначення, життєво важливим є надання засобів (механізмів), що вирішують питання довіри між об’єктами виробництва, проектувальниками та кінцевими користувачами. Проектувальники повинні мати гарантії щодо невикористання їхніх проектів з незаконною метою та збереження комерційної (технологічної) таємниці. А кінцеві споживачі повинні мати гарантії, що їхній пристрій не контролюється сторонніми особами та або не відбувається витоку вразливої інформації про цей пристрій.

Метою виявлення та захисту від троянських атак на апаратні засоби комп’ютерних систем є забезпечення того, що ніякі зловмисні електронні схеми не вбудовано в ці засоби. Існують рішення для захисту від троянських атак, що використовують архітектурні надбудови щоб унеможливити проникнення троянця в IP [85-87]. Однак, такий підхід базується, як вже було сказано вище, на порівняння "підозрілого" IP з еталонним.

Одним зі шляхів забезпечення того, що наявність в системі апаратного троянця не вплине на функціональність виробу є попередження негативного впливу троянця на функціонал ПЛІС. Варіант створення надійних комп’ютерних засобів пропонується в [88]. Запропоноване рішення повністю перебирає на себе відповідальність за використання всіх апаратних ресурсів обчислювального засобу. Інші підхід передбачає запобіганню виклику троянця [89]. Інші варіанти захисту від апаратних троянців базуються на тому що запускається копія програми на декількох обробляючих елементах [90]. В [91] використовуються так звані реконфігуровні логічні бар’єри у структурі проекту для запобігання активації та роботи апаратних троянців, що були додані на етапі виробництва мікросхеми.

Існує клас апаратних троянських схем, які розроблені так щоб активуватися спеціальним тригером. В результаті цього звичайні методи тестування не підходять для виявлення таких апаратних закладок через те, що вони не будуть активовані під час тестування такими методами.

Найсучасніші методи для вирішення проблеми апаратних троянців розроблені вже з урахування їх відстеження. Ці методи можна розділити на методології, що залежать від так званого side-channel (бічного каналу) та архітектурні методології [92].

Вплив апаратних троянців обмежується методологіями залежними від бічного каналу. Основна ідея цього методу полягає у спробі виявлення із великою ймовірністю присутності троянця шляхом відстеження параметрів мікросхеми в процесі перевантаження по різним параметрам схеми. Цими параметрами можуть бути затримка сигналу при проходженні по критичному шляху або споживання енергії у порівнянні з вільною від троянців мікросхемою. В роботі [93] спираються на локалізований аналіз струму для виявлення троянців - аналізується енергія з різних портів для виявлення троянця на енергії. В роботі [94] вивчали вплив троянця на перехідний струм, що споживається від джерела живлення, інтегральної схеми, використовуючи статистичні методи. В [95] використовували аналіз по методу затримки на критичного шляху для виявлення троянців. В [96] було запропоновано метод генерації тестових векторів, який може використовуватися для встановлення відмінностей між сигналами бічних каналів вільної від троянців і інфікованої троянцем мікросхеми.

Методології, що спираються на архітектурні рішення дозволяють збільшити ймовірності активації апаратного троянця під час тестування. Так підвищується активність троянців шляхом введення фіктивного тригера у проектне рішення. Обираються місця для розташування фіктивних тригерів, спираючись на граничне значення ймовірності переходу. Так в одному випадку використовується зміна рівнів напруги для збільшення енергетичних витрат схеми, із вбудованим апаратним троянцем. В іншому варіанті реалізації цієї методології захист всіх логічних елементів проекту обчислювального засобу реалізується за допомогою кільцевих генераторів. Присутність троянця виявляється через зміни, які він викликає у частоті

коливань кільцевого генератора який під'єднано до схеми через спеціальні логічні елементи [97-99].

Але всі перераховані методи мають одну суттєву проблему. Вони вимагають наявності еталонної, незараженої мікросхеми пристрою. Якщо розробник системи інтегрує в своє проектне рішення блоки інтелектуальної власності від сторонніх постачальників, ці методи стають недієвими.

2.4. Методи динамічного захисту блоків IP для реконфігуровних пристроїв.

В даному підрозділі представлено система захисту для IP, що вбудована у конфігураційні файли для ПЛІС [102]. Сторонами-учасниками є виробник комп'ютерних засобів на ПЛІС (ВКЗ, HM – hardware manufacturer), який проектує та виробляє ПЛІС пристрої, власник IP (ВІБ, IPO – intellectual property owner), який розробив нову логіку та системний інтегратор (CI, SI – system integrator), який буде використовувати IP , що надходять від його власника застосовуючи пристрої власника апаратного забезпечення. Тобто необхідно знайти рішення, що дають можливість обмежувати кількість використаних ПЛІС у проекті.

Розглянемо систему захисту, що заснована на використанні криптографічних примітивів для забезпечення необхідного рівня безпеки. Захист за допомогою симетричного шифрування проектного рішення та для забезпечення конфіденційності конфігурації (у деяких ПЛІС вже реалізована функція симетричного шифрування, що базується на відомих алгоритмах шифрування, таких як AES та 3DES [105-106]). Такі криптографічні функції можуть бути використані не тільки для забезпечення необхідного рівня конфіденційності: в конструкції CMAC (Cipher-based Message Authentification Code) [107] процес обчислення здебільшого ідентичний для того ж процесу у шифруванні за допомогою шифрування CBC – технологія блочного шифрування за допомогою побудови ацтентифікаційного коду

повідомлення з блочного шифру. Таким чином, для того, щоб зберегти відбиток одного крипто-компоненту малим за розміром, реалізація одного шифро-блоку може бути використана як для дешифрування, так і MAC-верифікації, використовуючи схему CBC в обох випадках. Таким чином, сторона, якій не відомі ключі не може розраховувати на отримання нового шифро-тексту, який би був прийнятий як достовірний. Так, два ключі можна вважати одним (хоча і довшим) ключем *k*. Аутентифіковане шифрування значення x з відкиданням тексту шифру (який включає значення MAC) для зворотного кроку дешифрування тексту шифру з перевіркою на помилку аутентифікації, що базується на значенні MAC, яке надходить як частина тексту шифру.

Використання CBC для конфіденційності разом з CMAC для аутентифікації є лише прикладом зручної у застосуванні схеми. Можна використовувати будь-яку підходящу симетричну криптографічну схему, що забезпечує аутенфіковане шифрування у відповідному контексті. Варто зазначити, що в конкретному випадку використання CBC з CMAC реалізація шифрування блочного шифру або дешифрування блочного шифру є достатньою у ПЛІС: можна використовувати блочний шифр "зворотно" для одного з двох криптографічних шляхів, тобто використовувати заснование на CMAC дешифрування блочного шифру замість етапу шифрування.

Асиметричне шифрування: якщо необхідно використовувати симетричне шифрування даних, виникає проблема узгодження загального ключа k між сторонами процесу (зазвичай, через ненадійний канал комунікації). Використовуючи асиметричну криптографію, пара ключів, що складаються з публічного та приватного компонентів, використовується для подолання ключових недоліків транспортування симетричних методів, ціною вищої складності системи та обчислень. Першим широковідомим прикладом шифрування з публічним ключем була схема Діфі-Хельмана (ДХ), що може бути використана для узгодження ключів для симетричного

шифрування. Правильне використання у зв'язці з функцією запозичення ключа (ФЗК, KDF – key derivation function), що базується на криптографічній хеш-функції, що є доволі елегантним рішенням. Важливий варіант цього являється схема ДХ з використанням криптографії еліптичної кривої, КЕДХ. Шифрування з публічним ключем також може використовуватися для аутентифікації через цифровий підпис.

Етап узгодження ключа

Для протокольних взаємодій, набір сторін визначимо як $Z = \{HM, IPO, SI, FPGA\}$, що відповідає сторонам, які описані вище, а також будь-який конкретний пристрій ПЛІС (FPGA) за правом вважається стороною. Ключ для симетричного шифрування, що обирається стороною $z \in Z$ позначається як K_z. Ключі для асиметричної шифрування надаються парами (PK_Z, SK_Z), де PK_Z – компонент публічного ключа (який може бутии відомий будь-кому) та SK_Z – приватний, або секретний компонент (в загальному випадку, має бути відомий лише для z, але іноді може бути поширений до інших сторін).

Схема узгодження ключа базується на схемі ДХ (включаючи КЕДХ) описана у [104]. Взявши публічні і приватні ключі $PK_Z, SK_Z, PK_{Z,}, SK_Z$, двох будь-яких сторін та додаткову бітову строку, яку позначимо як *OtherInfo*, у якій усвідомлюється, що ключі були сотворені на загальних параметрах домену, ці сторони можуть визначити матеріал симетричного ключа оцінивши ключ $(PK_Z, SK_{Z,}, OtherInfo)$ і ключ $(PK_{Z'}, SK_Z, OtherInfo)$, де *ключ* – це функція, що поєднує базові ДХ примітив (можливо, у варіації КЕДХ) з функцією запозичення ключа (ФЗК, KDF – key derivation function). Сторони використовують публічні ключі одна одної та свій власний секретний ключ як вхідні дані для ДХ-примітиву. З цього примітиву буде отримано однакові проміжні результати для обох сторін. Для отримання кінцевих даних, ФЗК застосовується для заданих вхідних даних: змінюючи значення строки

OtherInfo (яка стає частиною вхідних даних ФЗЧ) і можна застосувати алгоритм ДХ зі статичним ключем для отримання незалежних ключів.

Передумови та припущення

Для ефективної реалізації схеми захисту, припустимо наявність таких сторін взаємодії.

1. Довірена сторона. Загально довіреною стороною вважається виробник комп'ютерних засобів на ПЛІС (ВКЗ). Інші сторони, що беруть та не беруть участі у протоколі можуть вважати ВКЗ надійним та неупередженим, тобто ВКЗ ні поширить ніяких секретів, ні буде діяти нечесно на користь когось іншого. Всі інші сторони, однак, самі по собі вважаються не довіреними і можуть намагатися ввести в оману.
2. Надійна шифрування – приймається, що всі криптографічні примітиви є стійкими. Сюди входять симетричні та асиметричні криптографічні примітиви. Більш того, реалізації, що використані у протоколі приймаються як нечутливі до збоїв та стійкі до підробки, а також залишаються безпечними при атаках за бічним каналом.
3. Середовище захисту ПЛІС, що включає:
    a. Унікальний ідентифікатор пристрою (значення l-біта) присвоюється виробником апаратного забезпечення, що доступний з матриці.
    b. Симетричний ключ K_{HM} (значення m-біта) інтегрується ВКЗ під час процесу виробництва, який може бути прочитаний внутрішнім механізмом дешифрування, не доступний для зчитування через кристал ПЛІС.
    c. Сховище симетричних ключів K_{FPGA} (також, значення m-біта), що реалізується не енергонезалежна пам'ять і дозволяє зберігати змінний ключ. Ключ, що зберігається у K_{FPGA} може

бути оновлений використовуючи як зовнішній інтерфейс (наприклад, JTAG, SelectMap), або через внутрішній порт з реконфігуровної логіки ПЛІС. Однак, ключ лише може бути зчитаний за допомогою використання внутрішніх процедур дешифрування.

d. Регістр обміну даним, що може бути доступний через стандартизований інтерфейс конфігурації на кшталт JTAG, так і через реконфігуровну матрицю з використанням логіки подвійних портів. Ця особливість вже доступна на багатьох звичайних ПЛІС, на основі визначених користувачем інструкцій у протоколі JTAG.
e. Захищені від модифікацій компоненти управління та безпеки, що стійкі до атак на пристрй [108-111ы].

Етапи реалізації захисту IP

Пропонуються кроки, що виконуються кожною зі сторін відповідно до того, що всі сторони дотримуються єдиних правил, без будь-яких порушень. Існує п’ять головних кроків у такій схемі:

1. Налаштування. На етапі запуску нового класу пристроїв ПЛІС, ВКЗ створює специфічниу бітову послідовність для ПЛІС, модуль персоналізації (МП), що буде використаний у подальшому на етапі персоналізації. Зашифрована версія такого МП стає доступною для всіх учасників, разом з публічним ключем, що відноситься до нього.
2. Ліцензування. Коли власник IP пропонує ліцензії для CI, він надає свій власний публічний ключ. Всі ідентифікатори пристроїв ПЛІС на яких будуть використовуватися ліцензовані IP передаються власникам.

3. Персоналізація. МП використовується для того, щоб встановити специфічний апаратний ключ у кожен ПЛІС на якому він виконується.
4. Конфігурування. Використовуючи інформацію пристрою, ВІБ надсилає копії файлу конфігурації, що містять його IP, до СІ, які зашифровані конкретно для кожного ПЛІС.
5. Встановлення. СІ встановлює IP у кожен ПЛІС (використовуючи необхідну зашифровану копію).

Кроки 1-3 можуть вважатися одноразовими операціями (етапи ліцензування та конфігурації). Кроки 4 та 5 (частини етапів конфігурації та ліцензування) необхідно повторювати для кожного ПЛІС, який буде використовувати захищений IP.

Налаштування включає такі етапи:

a. ВКЗ генерує симетричний ключ K_{HM} та пару асиметричних ключів (SK_{HM}, PK_{HM}) для узгодження.
b. ВКЗ створює специфічний потік бітів P для ПЛІС, так, що P реалізує схему узгодження ключів. P включає приватний ключ SK_{HM}. Всі компоненти, що застосовуються, повинні бути стійкими до відмов та зовнішніх втручанн [106].
c. Після того, як створено файл бітового потоку P, він шифрується з використанням секретного ключа K_{HM} в зашифрований файл конфігурації $P_{enc} = enc_{K_{HM}}(P)$.
d. Секретний ключ K_{HM} поставляється з кожним пристроєм на ПЛІС.

Після виконання цих дій ВКЗ розповсюджує зашифрований бітовий потік P_{enc} та компонент публічного ключа PK_{HM} всім сторонам, що беруть участь у виробництві.

Етап ліцензування може вважатися першою взаємодією між ВІБ та СІ. Для використання зовнішнього IP, СІ укладає угоду з ВІБ (зазвичай, для ліцензування обсягу). Далі, необхідно виконати такі важливі два кроки:

a. ВІБ створює пару ключів (SK_{IPO}, PK_{IPO}) і надсилає публічний компонент PK_{IPO} системному інтегратору.
b. СІ надає ВІБ список значень ідентифікаторів пристроїв, для яких СІ має намір отримати ліцензію.

Персоналізація використовує зашифровану конфігурацію P_{enc} та узгодження ключів в ПЛІС і передбачає однократне встановлення ключів в кристал. Реалізується даний підхід за допомогою таких етапів:

a. Використовуючи програмний інтерфейс, ПЛІС конфігурується зашифрованим модулем персоналізації P_{enc}, який ВКЗ зробив доступним та дешифрується використовуючи статично інтегрований ключ K_{HM}.
b. Далі, регістр обміну даними ПЛІС завантажується публічним ключем PK_{IPO} через спільний інтерфейс (наприклад, JTAG), таким чином розпочинаючи обчислювальний процес.
c. Модуль персоналізації, щойно завантажений, визначає симетричний ключ $key(PK_{IPO}, SK_{HM}, ID)$ використовуючи схему узгодження ключа. З уього моменту, ПЛІС може дешифрувати послідовність, що зашифрована з використанням цього ключа.

Забезпечення безпеки фунцією узгодження ключа передбачає те, що необхідно знати або ключ SK_{HM}, або ключ SK_{IPO} для обчислення ключа, що зараз зберігається в K_{FPGA}. Включення ідентифікатора пристрою у вхідні дані ФЗК забезпечує те, що K_{FPGA} відрізнятиметься для різних напівфабрикатів ПЛІС.

Етап конфігування передбачає, що для кожного ідентифікатора пристрою ПЛІС, для якого СІ придбав ліцензію, передається відповідний

файл конфігурації, який може бути використаний лише на конкретному кристалі ПЛІС. Даний механізм дозволяє ВІБ легко відстежувати кількість ПЛІС, що були сконфігуровані для використання ліцензованих IP. ВІБ виконує наступні кроки для генерування конфігураційного файлу для конкретного ПЛІС:

a. ВІБ відновлює специфічний ключ K_{FPGA} використовуючи свій власний секретний ключ та публічний ключ ВКЗ:

$$K_{FPGA} = key(PK_{HM}, SK_{IPO}, ID)$$

b. ВІБ шифрує простий конфігураційний файл IP використовуючи секретний ключ, таким чином зв'язуючи IP з конкретним пристроєм IP:

$$IP_{enc,ID} = enc_{K_{FPGA}}(IP).$$

ВІБ передає цей зашифрований ключ СІ.

На етап встановлення СІ конфігурує ПЛІС з персоналізованим IP ($IP_{enc,ID}$). (СІ конфігурує флеш-пам'ять конкретного ПЛІС, що позначений ідентифікатором пристрою з $IP_{enc,ID}$ для роботи з пристроєм). Через те, що K_{FPGA} доступний у ПЛІС з етапу персоналізації, цей крок дозволяє ПЛІС використовувати конфігураційний біт файл ВІБ шляхом дешифрування $IP_{enc,ID}$.

Реалізації криптографічних елементів в модулі персоналізації були прийняті як нечутливі до збоїв та захищені від втручань. Це є обов'язковим, через те, що зловмисник може атакувати пристрій під час реалізації заходів безпеки на ньому. Таким чином, засоби протидії атак на пристрій мають перевіряти, як виконуються процедури при нетипових умовах, таких як підвищення напруги живлення, підвищення температури або підвищення тактової частота. Простим спсобом виявлення збоїв викликаних такими

умовами, є використання багатьох ідентичних компонентів в модулі персоналізації, які функціонують з різниими тактовими частотами та зсувом у часі. Коли завершено всі обчислення, результати порівнюються, щоб виявити помилки функціонуванням.

Особливості реалізації модуля персоналізації

Декілька алгоритмів з публічним ключем були успішно реалізовані в ПЛІС, наприклад алгоритми RSA та ДХ з низьким відбитком було запропоновано у [111-113] і вже доступні у готових до використання IP блоках таких виробників ПЛІС, як Altera.

Додаткові особливості ПЛІС

Додатковою вимогою для схеми захисту є доступ до ключа через кристалу ПЛІС за допомогою внутрішнього інтерфейсу, що доступний лише для запису.

Для необхідного рівня досягнення захисту від атак зловмисників, більшість ВКЗ можуть вжити заходів для приховування вразливих частин логіки від зчитувань та маніпуляцій, одним з яких є розподілення частин модулів безпеки по різним та частинам кристалу.

Для аутентифікованого шифрування необхідна апаратна реалізація в пристрої ПЛІС. Це можна виконати на основі одного шифро-блоку (AES чи 3DES). Аутентифіковане шифрування забезпечує цілісність бітової конфігурації, тобто користувач не може змінити зашифровану конфігурацію для отримання іншої (пов’язаної) конфігурації, що буде прийнята пристроєм ПЛІС та змінити частину логіки.

2.5. Висновки до розділу 2

Сучасний рівень оснащення зловмисників, які втручаються в intellectual property core, різноманіття їхнього арсеналу методів та засобів зламу, а також загрози правильному функціонуванню обчислювальних засобів на ПЛІС типу SEU спонукає розробників цих IP, користувачів та

компаній-виробників кристалів програмовної логіки на до розробки та впровадження методів й засобів захисту. Ці засоби допомагають клієнтам створювати проекти обчислювальних засобів на ПЛІС, які є не тільки захищеними, але й стійкими до клонування та підробки. Застосування шифрування та криптографічно стійкої аутентифікації дозволяє значно підвищити рівень захищеності проектів на ПЛІС та в поєднанні із використанням самоналаштовуючих чіпів eFUSE дозволяє створити нездоланний кордон для зловмисників. Наслідків від одиночних збоїв можна уникнути якщо використовувати наявні апаратні засоби, що вбудовані в ПЛІС для виявлення та корекції помилок.

Однією із важливіших задач забезпечення безпечної реалізації і використання засобів обчислювальної техніки та протидії атакам на апаратні ресурси є захист від апаратних закладок (апаратних троянців). Велике різноманіття сучасних методів виявлення та захисту свідчить про актуальність даної проблеми. Головним недоліком цих методів є необхідність мати в розпорядженні еталонну (незаражену апаратним троянцем) мікросхему із якою буде проводитись порівняння різних параметрів, що дозволить виявити закладку зловмисників.

На прикладі ефективної реалізації схеми захисту IP-блоків в проектах обчислювальних засобів на ПЛІС, показано суттєве покращення рівня захисту, що застосовно до сучаних ПЛІС з незначними внесеннями змін до набора функцій та архітектури пристрою.

РОЗДІЛ 3

ЗАСОБИ ВИЯВЛЕННЯ ЗАГРОЗ ТА ЗАХИСТУ БЛОКІВ ІНТЕЛЕКТУАЛЬНОЇ ВЛАСНОСТІ ВІД АПАРАТНИХ ТРОЯНЦІВ

3.1. Реалізація методу ідентифікації та захисту блоків інтелектуальної власності від апаратних троянців простим блокуванням

Пропонується розглянути модифікацію комплексного метода [100] для виявлення та захисту від апаратних троянців в комп'ютерних засобах на ПЛІС, що представлений такими етапами (складовими) як просте блокування (ПБ), мультиплексування виходів реконфігурованих варіантів (МВРВ), мультиплексування реконфігуровних виходів IP-блоків зі схемою виявлення троянців за допомогою циклічної перевірки надлишковості (МЦПН) та мультиваріантна реалізація (МВР).

Захист блоків інтелектуальної власності від апаратних троянців без їхнього виявлення.

Іноді розробники не можуть дозволити собі витрати на виявлення троянців. Отже, метою розробника стає відвертання витоку конфіденційної інформації без виявлення проникших троянців. Наведемо опис заходів, що відносяться до простого блокування. Даний підхід реалізуються за допомогою обфускації виходів IP, що містять троянців і зворотна обфускація відбувається тільки на входах IP-отримувача. Іншими словами просте блокування передбачає блокування виводів апаратних засобів, вражених троянцями шляхом умисного внесення помилки до порції даних, що відправляються із подальшим скануванням вмісту цих даних на вході приймача, як показано на рис. 3.1.

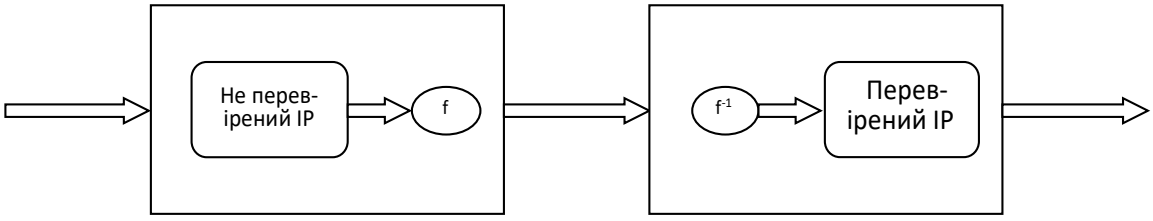


Рисунок 3.1 – Реалізація етапу простого блокування

Ця методологія використовується для уникнення витоку секретних даних через так звану позасхемну передачу. Для цього використовуються спеціальні "легковажні" функції для умисного перетворення, що дозволяє уникнути значних апаратних витрат. Для отримання вихідних даних відбувається зворотне перетворення, а потім данні передаються на вхід IP-приймача. Як функція перетворення пропонується проста функція, яка базується на операціях XOR, XNOR. На рис. 3.2 та рис. 3.3 наведено приклад перетворень простої функції перетворення та оберненої до неї функції.

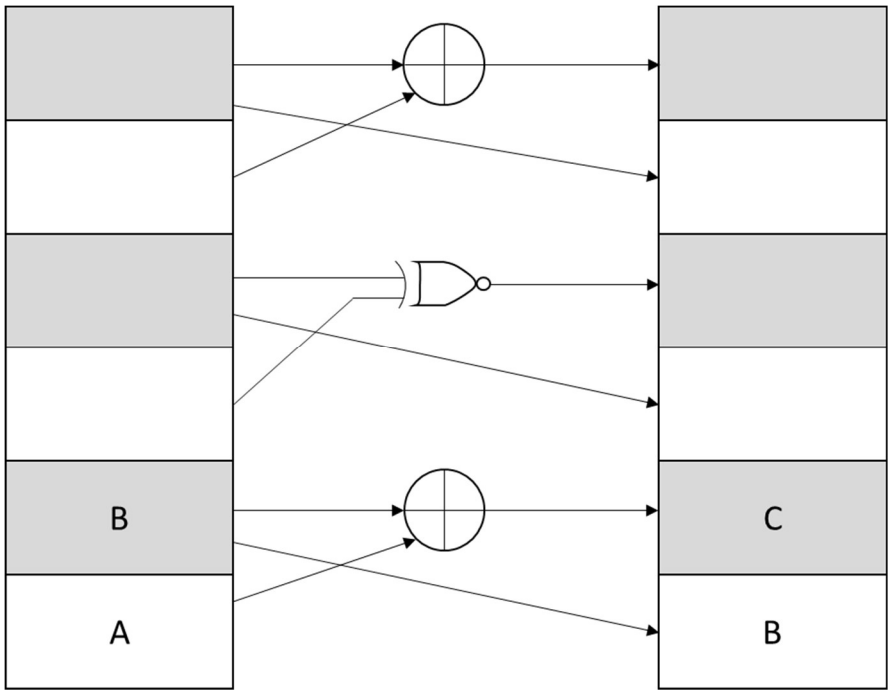


Рисунок 3.2 – Приклад перетворень простої функції

Обробка даних проводиться для кожних двох бітів за допомогою операцій XOR та XNOR та передачі значення першого біту іншому. Як показано на рис. 3.1, вихідні дані містять біти *A* і *B*, які будуть конвертовані операцією XOR у *C*. Далі *B* буде переміщене на місце *A*. Дія зворотної функції показана на рис. 3.3, де значення *B* і *C* конвертуються операцією XOR для повернення *A*. Далі *B* переміщується назад у початкове положення. Те саме буде зроблено для третього і четвертого бітів, але за допомогою операції XNOR. Якщо розмір вихідних даних непарний, останній біт може бути поміняний місцями з одним із інших останніх бітів. Для того, щоб ускладнити зловмисникам розкриття алгоритму

перетворень функцій пропонується виконувати часткову реконфігурацію ПЛІС для вибіркової замін функцій новими.

Для перевірки ефективності захисту ПБ від апаратних троянців реалізовано шість IP, які взято із відкритих джерел [100], що виконують відповідні перетворення (таблиця 3.1).

Таблиця 3.1 – Опис функцій, що реалізують IP

Назва	Перетворення
Solomon	Декодер Соломона Ріда з 204- та 188-байтовим входом та виходом і довжнами вихідних слів, відповідно
PID	Цифровий Пропорційно-інтегрально-диференціальний пристрій.
FPU	Набір операцій з плаваючо комою.
FHT	Швидке Перетворення Хадамхарда (ШПХ) 8-бітних вхідних даних з використанням матричного додавання.
F3M	$Y^3 \bmod(x^{97}+x^{12}+2)$, де Y - 194-бітний вхідний порт.
CA PRNG	Одновимірний двійковий автомат з оберненням країв, наприклад кільце.

Використано пристрій XC6SLX150-2FGG44T та Xilinx ISE v.13 для оцінки експериментальних результатів. Також, виконано оцінку величини потужності споживання із використанням генератора тактових сигналів на 100 МГц, крім блоку FPU, для якого було використано генератор на 10 МГц. Також, в результаті синтезу апаратних структур отримано дані про кількість використаних LUT, критичні затримки та потужність споживання. В таблиці 3.2 наведено результати синтезу для метода ПБ.

Таблиця 3.2 – Результати синтезу

Метод оцінки	Використані LUT	Затримка (нс)	Споживана енергія
FHT	24	1,701	1,278
PID	1332	3,912	1,410
F3M	148	1,286	2,378
FPU	9585	99,459	1,374
CA PRNG	110	1,834	1,380
Solomon	4974	4,429	1,374

Використано функцію обфускації для захисту виходів тестів від троянців, які вбудовуються в IP від третіх постачальників.. В таблиці 3.3 наведено підсумкові результати після застосування функції обфускації. Якщо порівняти таблицю 3.2. та таблицю 3.3, то отримуємо середній приріст використання LUT у 12.69%. Також, встановлено, що використання порівняльного тесту F3M з запропонованою функцією обфускації призводить до більших затримок ніж інші порівняльні тести через те, що розмір порту виходу дорівнює 194 біти і, відповідно, необхідно більше часу на обробку. З вищезазначеного можна встановити, що середнє збільшення для реалізованих тестів затримки критичного шляху складає 2.106% а середнє збільшення споживаної енергії - 0.64%.

Таблиця 3.3 – Результати синтезу після проведення обфускації

Метод оцінки	**Приріст відсотку використаних LUT**	**Приріст відсотку затримки**	**Приріст відсотку споживаної енергії**
FHT	0,53	0,50	0,832
PID	3,85	0,005	0,541
F3M	39,32	12,18	1,943
FPU	0,73	0,0096	0,08
CA PRNG	31.31	0.42	1.104
Solomon	0.45	0.0013	0.067
Середнє	12.69	2.106	0.64

3.2. Реалізація методу ідентифікації та захисту блоків інтелектуальної власності від апаратних троянців шляхом мультиплексування виходів реконфігурованих варіантів (MBPB)

У методі MBPB запропоновано посилити захист від атак троянців, зменшуючи потік витоку конфіденційної інформації, та зменшення кількості несправностей внаслідок атак троянців. Для уього необхідно використовувати декілька варіантів реалізацій IP від різних сторонніх постачальників. Таким

чином, всі виходи обчислювального засобу складаються з міксу виходів всіх IP, які інтегровано в систему (в тому числі ті, які реалізують однакові функції) [23].

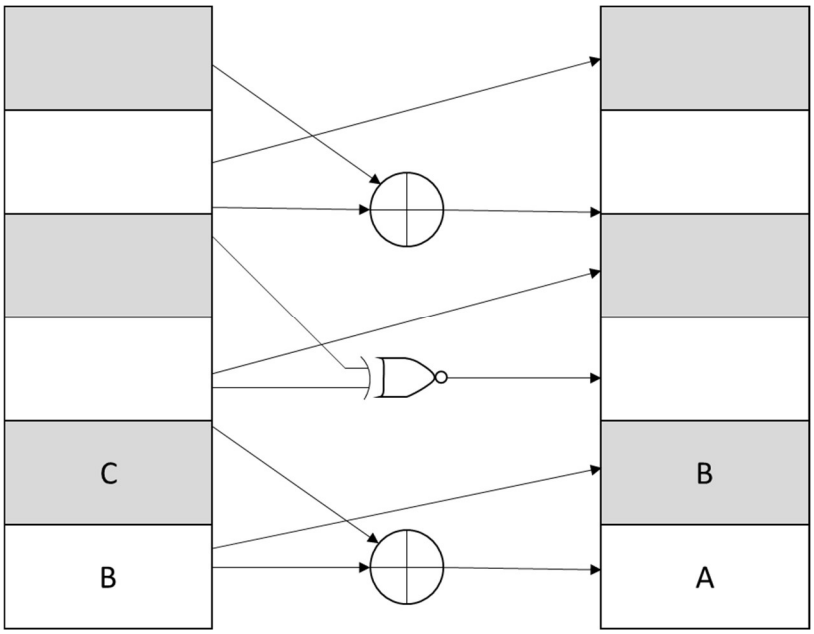


Рис. 3.3 – Зворотній порядок прикладу дезорієнтуючої функції з рис. 3 для повернення вихідних даних

Основна ідея, у випадку відсутності повністю надійного IP, полягає у використанні *m* екземплярів IP від різних сторонніх постачальників. Під час роботи вихід обирається випадково з *m* екземплярів IP. Таким чином, троянець буде впливати на вихід тільки тоді, якщо він активується поки обрано інфікований вхід IP. Крім того, пропонується періодично чи навіть випадково частково реконфігурувати ПЛІС, щоб замінити IP, що знаходяться у роботі, для того, щоб знизити можливості втручання у систему. Рис.3.5 ілюструє даний підхід. На заміну IP йде близько 8% часу, необхідного для програмування всієї схеми на FPGA, як це буде пояснено нижче.

Пропонуються такі критерії відбору виходів з різних IP:

1) Неупереджена випадкова вибірка де використовується генератор псевдовипадкових величин для відбору виходу з одного з IP. Таким чином ймовірність вибору IP однакова для всіх кандидатів відбору. Ця методологія може бути використана, якщо не можна обчислити ймовірність зараження троянцем.

2) Упереджена випадкова вибірка – проводиться зважене голосування серед наявних IP, які використовуються, для вибору потрібного виходу. Вага є обернено пропорційною ймовірності зараження IP троянцем. Вага може присвоюватись одним із таких способів:

- Можливе проведення відстеження поведінки IP для присвоєння вагових коефіцієнтів кожному з IP. Результат порівняння інформації на виходах функціонючих IP може використовуватись для присвоєння ваги під час роботи. Обчислення і корекція контрольної суми (CRC) на виходах IP також може бути використана як основа для побудови так званої «кривої засвоєння» і присвоєння ваги.
- Вага може бути присвоєна на основі відомостей про блок.

Витрати логічних ресурсів ПЛІС для реалізації методу МВРВ

Ефективність застосування МВРВ можна провести за допомогою оцінки кількості використаних LUT, рівня споживання електроенергії, часу часткової реконфігурації (ЧР). Для проведення експерименту застосовано плату з ПЛІС, що підтримує часткову реконфігурацію. В експерименті з багаторазовою реалізацією одного й того самого IP, використано модуль АЛП (ALU) та модуль блоку послідовного передавача (RS_TX). Кожен із цих блоків реалізовано декількома варіантами IP з різних джерел і проведено їх інтеграцію зі схемою ущільнення для побудови цілісної системи. Таблиця 3.4 ілюструє варіанти реалізації IP типу RS_TX без використання часткової реконфігурації (ЧР, PR). Величина споживаної електроенергії у таблиці 3.4 не включає енергію витоку плати.

Таблиця 3.4 – Деталі апаратної реалізації використаних ядер RS_TX без використання PR.

	Використані LUT	**Споживання енергії (мВт)**
RS_TX_n1	21	0.798

RS_TX_n2	27	0.823
RS_TX_n3	34	0.889

Вивчено ефект використання ЧР на показники споживання електроенергії і на число використаних LUT. У таблиці 3.5 показано приріст у відсотках у кількості LUT та споживаної енергії при задаванні реконфігуроіної зони для кожного IP. RS_TX_1PR_nl відображає використання лише однієї реконфігуровної зони для RS_TX_nl. Таблиця показує, що використання реконфігуровної зони збільшує необхідну кількість LUT. Якщо порівняти результати з таблиці 3.4 та таблиці 3.5, будуть спостерігатися витрати на використання ЧР для єдиного запрограмованого IP. Середній приріст у кількості LUT становить 31.53% і середній приріст у споживаній енергії становить 1.2733%.

Таблиця 3.5 – Деталі використання апаратної реалізації використаних ядер RS_TX з використанням ЧР.

	Приріст відсотку використаних LUT	**Приріст відсотку споживаної енергії**
RS_TX_1PR_n1	56.01	1.31

Таблиця 3.5 (продовження)

RS_TX_1PR_n2	27.16	1.42
RS_TX_1PR_n3	11.91	1.21
Середнє	31.69	1.3133

Вивчено ефект мультиплексування двох IP без ЧР. Таблиця 3.6 ілюструє номер варіанту LUT та споживану енергію, якщо використовується два IP в одному проекті з мультиплексором та без ЧР. RS_TX_T_nl2 показує комбінацію RS_TX_nl та RS_TX_n2. Якщо порівняти результати в таблиці 3.4 та таблиці 3.6, можна відмітити приріст у зоні розташування, що є очікуваним з огляду на

завантаження двох проектів на ПЛІС. Цей приріст пропорційний до розміру двох вибраних IP.

Таблиця 3.6 - Деталі апаратної реалізації ущільнення ядер RS_TX без використання ЧР.

	Використані LUT	**Споживання енергії (мВт)**
RS_TX_nln1	51	1.245
RS_TX_nln2	50	1.238
RS_TX_nln3	59	1.330

Окрім того, досліджено ефект ущільнення двох IP блоків (RS_TX) у випадку зберігання реконфігуровної зони для кожного IP. У таблиці 3.7 показвно кількість LUT, споживану електроенергію та час перемикання при використанні двох реконфігуровних зон для яких кожен із IP може бути замінений іншим. Як RS_TX_2PR_nln2 позначено використання блоків RS_TX_nl та RS_TX_n2 одночасно, для кожної індивідуальної реконфігуровної зони. Результати у таблиці 3.7 показують, що використання двох реконфігуровних зон збільшує кількість необхідних LUT Якщо порівняти результати в таблиці 3.6 та таблиці 3.7, знайдемо, що використання двох реконфігурованих зон з ЧР потребує більше LUT ніж змішування двох IP без ЧР. Однак, в ЧР ті два модулі можуть легко бути замінені *m* модулями.

Таблиця 3.7 – Деталі апаратної реалізації ущільнення ядер RS_TX з використанням ЧР.

	Використані LUT	**Споживання енергії (мВт)**
RS_TX-2PR_nln2	65	1.94
RS_TX-2PR_nln3	64	1.83
RS_TX-2R_n2n3	75	1.96

Через те, що при ЧР відбувається перемикання між IP, час на заміну одного варіанту коливається між 400 та 500 мсек, середнє значення 450 мсек. В той же час, тривалість програмування конфігурації в цілому займає від 5 до 6 секунд. В результаті, ЧР ефективна при зміні конфігурації (часом при цьому нехтується) у порівнянні з часом на програмування конфігурації в цілому.

Те ж саме проведено з використанням IP-блоку АЛП (ALU) від різних виробників. В таблиці 3.8 показано кількість використаних LUT та рівень споживання електроенергії, якщо використовувати кожен IP окремо.

Таблиця 3.8 – Деталі апаратної реалізації використаних ядер ALU без використання ЧР.

	Використані LUT	**Споживання енергії (мВт)**
ALU_n1	8	0.173
ALU_n2	10	0.236
ALU_n3	5	0.1123

В таблиці 3.9 наведено кількість LUT і рівень споживання елетроенергії у разі мультиплексування двох ALU без ЧР. Блок ALU_nln2 показує комбінацію першої та другої реалізації ядра логічного пристрою (ALU_nl та ALU_n2).

Таблиця 3.9 – Деталі апаратної реалізації ущільнення двох ядер ALU без використання PR.

	Використані LUT	**Споживання енергії (мВт)**
ALU_nln2	17	0.269
ALU_nln3	12	0.185
ALU_n2n3	19	0.298

У Таблиці 3.10 міститься інформація, щодо кількості LUT, рівня споживаної електроенергії та часу перемикання, якщо резервувати реконфігуровану зону для кожного ALU IP.

Таблиця 3.10 – Деталі апаратної реалізації ущільнення двох ядер ALU з використанням ЧР.

	Використані LUT	**Споживання енергії (мВт)**
ALU-2PR_nln2	45	0.48
ALU-2PR_nln3	36	0.31
ALU-2PR_n2n3	38	0.3932

3.3. Реалізація методу ідентифікації та захисту блоків інтелектуальної власності від апаратних троянців шляхом мултиплексування реконфігурованих виходів IP-блоків зі схемою виявлення троянців

Для подальшого зниження ймовірності інформаційних втрат, пропонується використовувати функцію часткової реконфігурації ПЛІС, яка ділить логічне рішення (ПЛІС) на два різних типи: реконфігурована логіка та статична логіка. Часткова реконфігурація дозволяє модифікувати області зі змінюваною конфігурацією у ПЛІС без порушення, не піддаючи небезпеці цілісність додатків, що виконуються у частині статичної логіки, як проілюстровано на рис. 3.7. У даній методології кожне IP-ядро резервує область логіки зі змінюваною конфігурацією, у той час як динамічний блок детектора троянців і блоку обробки аварійних сигналів резервують статичну область логічних ресурсів.

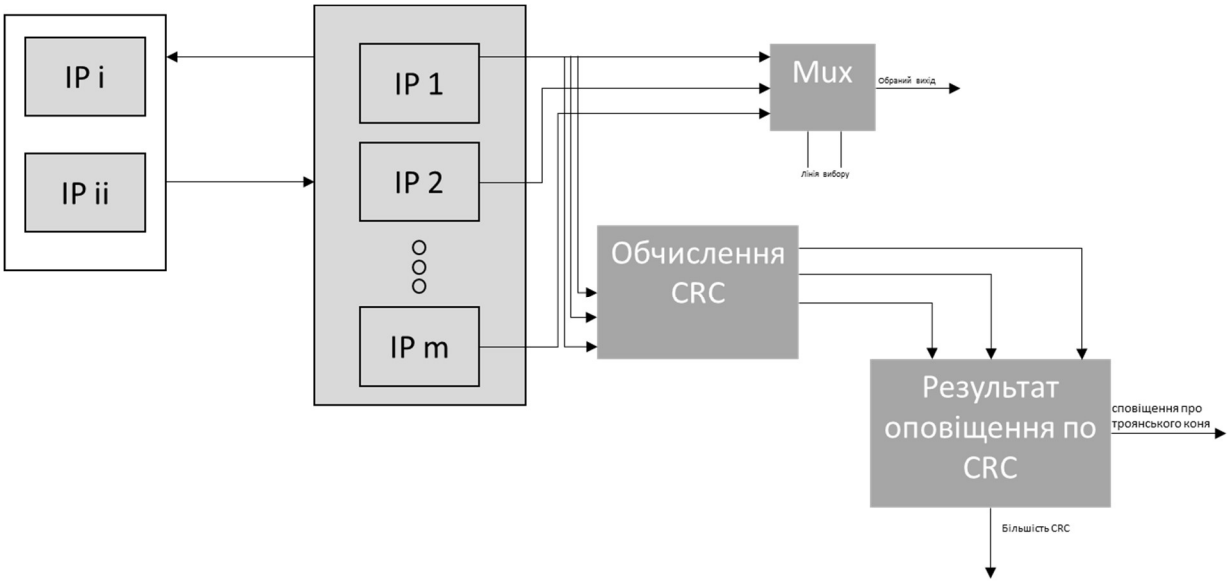


Рис. 3.5 – Мультиплексування виходів IP зі змінюваною конфігурацією для вибору вірного і додавання частини CRC для перевірки виходу.

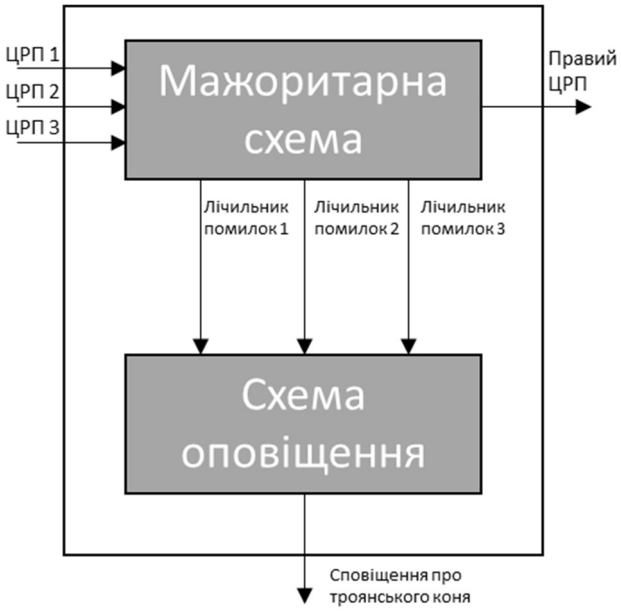


Рис. 3.6 – Схема голосування CRC.

Підрахунок числа помилок показує кількість разів, коли CRC кожного IP несправна.

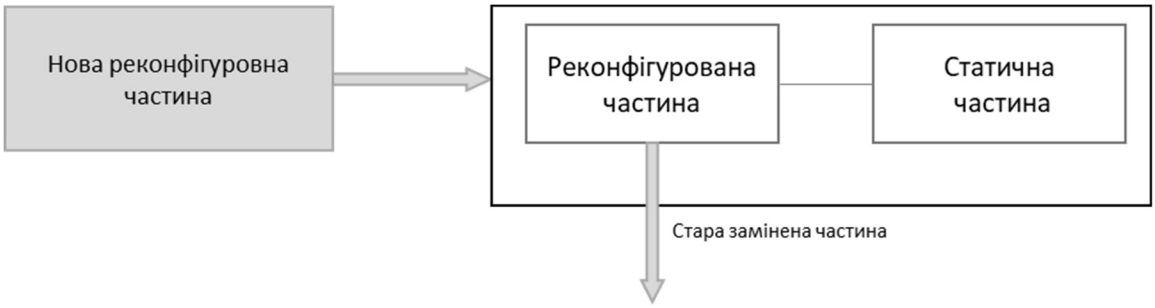


Рис. 3.7 – Концепція часткової реконфігурації FPGA

Використовуючи функцію ЧР ПЛІС, можна оптимізувати мультиплексування для забезпечення більшого захисту від троянців для обчислювальних засобів на ПЛІС. Під кожен IP-блок резервується область зі змінюваною логікою, в той час, як блок мультиплексор та схема голосування CRC резервуватиме статичну область логіки. Щоб зменшити ймовірність витоку інформації, що представлє собою комерційну таємницю, пропонується виконувати часткову реконфігурацію ПЛІС, щоб періодично замінювати поточний IP на новий. Для цього необхідно утворити чергу із IP, для забезпечення функціонування деяких з них у поточний момент часу, поки інші блоки інтелектуальної власності чекатимуть завантаження у черзі.

Алгоритм 1 описує метод ущільнення реконфігурованих IP-виходів для сторонніх блоків, використовуваних на ПЛІС. Розробник системи має *m* реалізацій підозрілого IP від різних постачальників. ПЛІС конфігурується з використанням тільки 3 блоків (для прикладу), мультиплексора і схеми голосування CRC. Система працює нормально і вихід обирається шляхом мультиплексування виходів різних варіантів. Як тільки схема голосування CRC виявляє помилку в IP, її лічильник числа помилок збільшиться. Якщо він перевищить обране порогове значення, даний IP він буде позначений як інфікований і буде замінений на новий блок. Аномальним буде вважатися блок, для якого число помилок перевищує деяке встановлене порогове значення, і буде видалятись із системи за допомогою проведення ЧР.

Після певного періоду часу одне з трьох ядер замінюється першим із IP із черги. Крім того, результати обчислення і коригування контрольної суми можуть

використовуватись для оновлення ваги даного IP у процесі мультиплексування [24].

Алгоритм 1, який виконує мультиплексування даних із виходів реконфігурованих IP та реалізація схеми виявлення троянців .

Алгоритм 1. Ущільнення реконфігурованих виходів IP і схема виявлення троянців CRC.

Етап 1. Виконується завантаження IP з трьох різних джерел.

Етап 2. Система запускається у звичайному режимі роботи.

Етап 3. Присвоюється значення 0 змінній i.

Етап 4. При нормальному виконанні перейти до етапу 9.

Етап 5. Якщо сталася помилка ЦРП виконати збільшення лічильника помилок.

Етап 6. Якщо лічильник помилок перевищує допустимий поріг, необхідно виконати часткову реконфігурацію ПЛІС для повного вилучення зараженого IP.

Етап 7. Замінити заражений IP найближчим у черзі.

Етап 8. Позначити аномальний блок як заражений блок.

Етап 9. Якщо стік час циклу, використати часткову реконфігурацію для заміни числа і IP першим блоком у черзі.

Етап 10. Присвоїти і значення $(i + 1) mod 3$.

Етап 11. Алгоритм закінчено.

Витрати логічних ресурсів ПЛІС для реалізації методу MCRC.

Для експерименту використано три блоки IP UART як множину варіантів. Обчислення CRC виконується на кожному виході UART. Зазначимо, що використання обчислення CRC в даному випадку може не виконуватися, через те, що UART зазвичай має функцію обчислення CRC для відстежування помилок відправки.. У таблиці3.11 наведено деталі синтезу використовуючи пристрій Virtex 6.

Таблиця 3.11 – Деталі апаратної реалізації методу MCRC на ядрах UART.

	Використані LUT	Затримка (нс)	Споживання енергії (мВт)
UART1	29	1.911	75
UART2	25	1.634	5
UART3	66	2.354	11
FinalUART	168	2.634	110

Те ж саме проведено з IP ALU. У таблиці 3.12 наведена детальна інформація про синтез з використанням ПЛІС серії Virtex 6. Енергія витоку дорівнює 1.293 мВт і є однаковою для всіх варіантів.

Таблиця 3.12 – Деталі апаратної реалізації методу MCRC на ядрах UART.

	Використані LUT	Затримка (нс)	Споживання енергії (мВт)
ALU1	48	12.22	2
ALU2	36	4.623	13
ALU3	169	3.767	50
FinalALU	221	12.528	101

3.4. Реалізація методу ідентифікації та захисту блоків інтелектуальної власності від апаратних троянців шляхом використання мультиваріантної реалізації

Для забезпечення необхідного рівня захисту від апаратних троянців, пропонується метод мультиваріантної реалізаці для якої пропонується додавання деякого кількості IP-блоків від різних постачальників і використання динамічного блоку виявлення троянців для визначення підозрілого IP. Рис.3.8 демонструє, як працює схема. Порівнюється вихід *m* ядер, і якщо є невідповідність між різними виходами, існує ймовірність присутності троянеця. Якщо є три чи більше IP, можна використовувати схему голосування, яка обирає правильний вихід і, при необхідності, активує сповіщення про наявність троянця у системі [101].

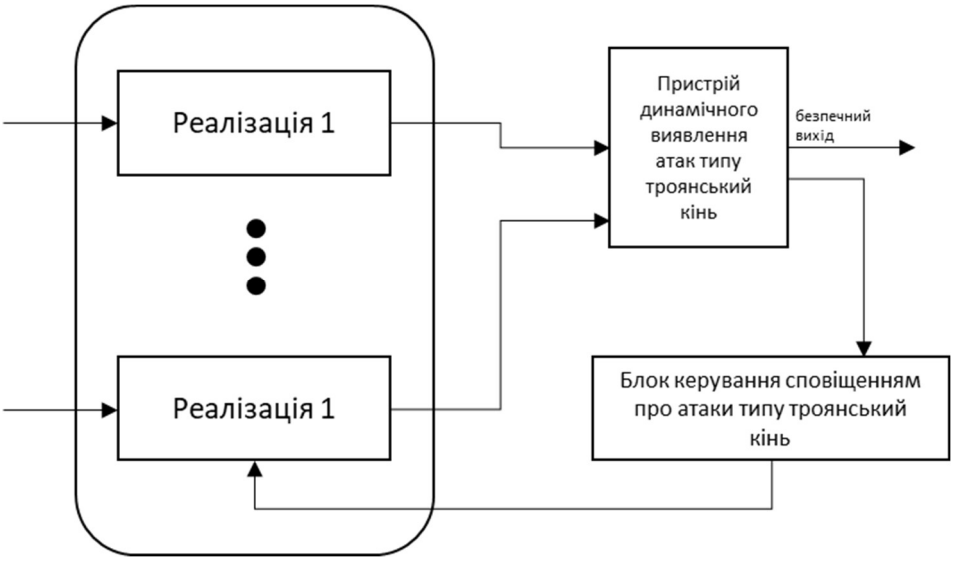


Рис. 3.8 – Використання множини варіантів для відстеження Троянця

Блок управління оповіщеннями присутності троянця може частково реконфігурувати ПЛІС щоб вилучити заражений блок і замінити його на новий поки інша частина логіки ПЛІС працює. Динамічний детектор троянців також позначить невідомого виробника блоків як підозрілого виробника.

Розробник проектів обчислювальних засобів на ПЛІС зберігає *m* різних реалізацій підозрюваного блока від різних виробників. ПЛІС конфігурується, наприклад, використовуючи тільки три блока і схему сповіщення про троянців. Система працює нормально і вихідний сигнал вибирається шляхом вибору мажоритарною схемою з різних варіантів. Коли відбувається оповіщення про наявність троянця в системі, визначається підозрілий блок. Даний блок замінюється на інший не підозрілий. Варто відзначити, що сповіщення про наявнсть троянця відбувається також якщо лічильник помилок IP перевищує допустимий поріг.

Витрати логічних ресурвсів ПЛІС для реалізації методу МВР

Для проведення синтезу, використано всі три IP UART як множину варіантів і їх інтегровано зі схемою динамічного виявлення Троянців для побудови FinalUART. У таблиці 3.13 показано деталі синтезу з використанням пристрою Virtex 6.

Визначено, що затримка FinalUART дорівнює суммі значень максимальної затримки ядра UART та затримки схеми динамічного виявлення троянців. Енергія витоку дорівнює 1.293 мВт і є однаковою для всіх випадків.

Таблиця 3.13 – Деталі апаратної реалізації методу MV на ядрах UART.

	Використані LUT	**Затримка (нс)**	**Споживання енергії (мВт)**
UART1	25	1.882	15
UART2	26	1.643	8
UART3	63	2.345	12
FinalUART	125	2.364	20

Останній експеримент повторюється, але з використанням ALU IP від різних виробників. В Таблиці 3.14 показано результати з використанням пристрою Virtex 6. Затримка FinalALU дорівнює сумі значень максимальної затримки ядра ALU та затримки схеми динамічного виявлення троянців. Затримка схеми динамічного виявлення для даного випадку дорівнює 1.695 наносекунд. Енергія витоку дорівнює 1.293 мВт і однакова для всіх випадків.

Таблиця 3.14 – Деталі апаратної реалізації методу MV на ядрах ALU.

	Використані LUT	**Затримка (нс)**	**Споживання енергії (мВт)**
ALU1	50	12.01	3
ALU2	38	4.610	11
ALU3	172	3.750	48
FinalALU	187	13.90	104

Такий же експеримент проведено з використанням IP, що представляє собою шифроблоки AES, що може обробляти блоки даних по 128 біт, використовуючи 128-бітний ключ. Таблиця 3.15 показує деталі синтезу з використанням ПЛІС серії Virtex 6.

Зазначимо, що затримка FinalAES дорівнює сумі значень максимальної затримки ядра AES та затримки схеми динамічного виявлення троянців.

Затримка схеми виявлення у цьому експерименті дорівнює 0.214 наносекунд. Енергія витоку дорівнює 3.769 мВт і однакова для всіх випадків.

Таблиця 3.15 – Методологія множини варіантів AES.

	Використані LUT	**Затримка (нс)**	**Споживання енергії (мВт)**
AES1	700	4.336	468
AES2	1560	5.218	138

Таблиця 3.15 (продовження)

AES3	261	2.981	422
FinalAES	1150	5.72	1055

3.5. Висновки до розділу 3.

В даному розділі продставлено модифікований комплексний метод, що базується на виявленні троянців та захисті системи від атак такого типу, без реалізації алгоритмів виявлення троянців на етапі тестування. Запропоновані методи працюють під час виконання програми замість традиційного тестового етапу. Дана робота розглядає методи захисту від апаратних закладок, що можуть бути вбудовані в сторонні IP, де не доступне еталонне рішення. Запропоновано методи для простого захисту на рівні системи та більш складні методи, що забезпечують виявлення та вилучення троянця з системи. Метод ПБ запропоновано як першу схему простого захисту, що базується на обфускації виходу підозрюваного IP блоку для приховування будь-якого витоку інформації. Другий метод простого захисту, що запропонований у роботі, носить назву МВРВ, який захищає систему від витоку конфіденційної інформації та несправностей. Також було представлено два інші більш складні методи, МЦПН та МВР для динамічного усунення заражених троянцями IP-блоків та повідомлення про атаки типу троянський кінь. Вони використовують переваги сучасних ПЛІС для очистки заражених IP-блоків від троянців.

ВИСНОВКИ

В магістерській дисертації проаналізовано архітектурно-структурну організацію елементно-компонентної бази, тобто ПЛІС та сучасні напрямки розвитку ПЛІС-технологій. Окреслено проблеми створення обчислювальних засобів на їхній основі та проведено класифікацію загроз для проектів на ПЛІС, що дозволило наочно висвітлити основне завдання дослідження - підвищення ефективності засобів захисту блоків інтелектуальної власності проектів цифрових обчислювальних засобів на ПЛІС шляхом відображення завдань, методів і алгоритмів на архітектуру і структуру цих засобів.

В основу магістерської роботи покладено комплексний метод вирішення задачі захисту IP-блоків від такого типу атак як апаратні троянці. Незважаючи на існування ефективних методів вірішення поставленої задачі, більшість існуючих рішень мають основним недоліком те, що необхідно мати не заражену троянцем копію IP. Предствлений модифікований комплексний метод дозволяє забезпечити захист на рівні системи, а, також, виявляти та вилучати апартні закладки в процесі функціонування прострою, що відповідає поставленим задачам дослідження. Іншим, дуже важливим, показником є додаткове споживання електроенергії системами захисту в комп'ютерних засобах на ПЛІС. В деяких випадках можливе настільки значне її додаткове споживання, що може привести до того, що розвобники відмовляться від такої системи захисту. Результати дослідження запропонованого методу показали, що додаткове енергоспоживання не перевищує 2%, у той час, як інші методи додають більшого енергоспоживання до проекту.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Развитие ускорителей специализированных вычислений [Текст] / А.А. Якуба, Э.И. Комухаев, С.Г. Рябчун // Математичні машини і системи , 2010, N2 с.10-20

2. Asic design starts decline, but revenues hold up – Gartner [Електронний ресурс] -2013.- Режим доступу: http://www.electronicsweekly.com/news/manufacturing/asic-design-starts-decline-but-revenues-hold-up-gartner-2012-03

3. Expect a Breakthrough Advantage in Next-Generation FPGAs [Електронний ресурс] /Altera corporation – White Paper. – 2013 – Режим доступу: http://www.altera.com/literature/wp/wp-01199-next-generation-FPGAs.pdf

4. Опанасенко В. ПЛІС типа FPGA фирмы Xilinx: возможности, проектирование и применение / В. Опанасенко, В. Сахарин // Электронные компоненты и системы. – 2003. - № 4. – С. 7-11.

5.Построение высокоскоростных сетевых систем сбора данных и управления на основе ПЛІС фирмы Xilinx / А. Бритов, А. Макеенок, А. Сотников, С. Хлебников // Chip NEWS Украина. – 2006. – № 3. – С. 54-56.

6. Тарасов И. Возможности FPGA фирмы Xilinx для цифровой обработки сигналов / И. Тарасов // Компоненты и технологии. – 2007. - № 5. – С. 68-74.

7. Проектирование цифровых устройств на кристаллах ПЛІС с использованием Core-ядер / А.В. Палагин, В.Н Опанасенко, В.Г. Сахарин, А.А. Софиюк //Материалы 10-й международной конференции по автоматическому управлению "Автоматика-2003".-Т.3.-Севастополь, 2003. С.97-98.

8. Соловьев В. В. Проектирование цифровых систем на основе программируемых логических интегральных схем. –М.: Горячая линия – Телеком, 2001. – 636 с.

9. Потехин Д. С., Тарасов И. Е. Разработка систем цифровой обработки информации на базе ПЛІС. М.: Горячая линия – Телеком, 2007

10. Tabula and Algo-Logic Collaborate to Release Second Generation Ternary Search Engine. [Електронний ресурс] / Design And Reuse. 2013. - Режим доступу: http://www.design-reuse.com/news/32366/tabula-algo-logic-ternary-search-engine.html?utm_content=147735&utm_campaign=32366&utm_medium=socnewsalert&utm_source=designreuse

11. Scott Hauck, Andre DeHon. Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation / Morgan Kaufmann, 2008. - 944 p.

12. Палагин А.В. Реконфигурируемые вычислительные системы / А.В. Палагин, В.Н. Опанасенко // - Киев: Просвіта, 2006. - 295 с.

13. Палагин А.В. Реконфигурируемые структуры на ПЛІС / А.В. Палагин, В.Н. Опанасенко, В.Г. Сахарин // УСиМ. - 2000. - № 3. - С. 32-39.

14.Опанасенко В.М. Архітектурна організація реконфігуровних комп'ютерів на базі ПЛІС / В.М. Опанасенко, І.Г. Тимошенко // Радіоелектроніка. Інформатика. Управління. - Запоріжжя: ЗНТУ, 2004. - № 2(12). - С 139-144.

15. Палагин А.В. Проектирование реконфигурируемых систем на ПЛІС / А.В. Палагин, В.Н. Опанасенко, А.Н. Лисовый // Тр. 7-й междунар. науч.-практ. конф. «Современные информационные и электронные технологии», 22-26 мая 2006, Одесса.-1.-С. 164.

16. Проблеми побудови частково реконфігурованих систем на ПЛІС / Р. Б. Дунець, Д. Я. Тиханський // Радіоелектрон. і комп'ют. системи. - 2010. - № 7. - С. 200-204.

17.Dynamic partial reconfiguration on FPGA[Електронний ресурс]. – Режим доступу:http://www.vlsi-world.com/content/view/48/47. *–17.03.2011*

18.Gokhale, M.B. Reconfigurable Computing: AcceleratingComputation with Field-Programmable Gate Arrays [Text] / M.B. Gokhale, P.S. Graham // Springer.– 2005.

19. Хаханов В.И., Литвинова Е.И., Гузь О.А. Проектирование и теститрование цифровых систем на кристаллах. – Харьков: Изд-во ХНУРЭ, 2009, -484 с

20. Немудров В., Мартин Г. Системы на кристалле. Проектирование и развитие. — М.: Техносфера, 2004, с. 216.

21.Шагурин И., Шалтырев В., Волов А. «Большие» FPGA как элементная база для реализации систем на кристалле//Электронные компоненты, 2006, №5, с.83—88.

22. D. McGrath Report: Semiconductor IP market to double in five years. [Електронний ресурс] / EETIMES – 2012 – Режим доступу: http://www.eetimes.com/document.asp?doc_id=1261490

23. Кузелин М.О. Современные семейства ПЛИС фирмы XILINX / М.О. Кузелин, Д.А. Кнышев, В.Ю. Зотов –М.: "Горячая линия – Телеком", 2004, 440 с

24. All Programmable Technologies and Devices [Електронний ресурс] / Xilinx Inc. -2012.- Режим доступу: http://www.xilinx.com/about/company-overview/index.htm

25. 7 Series FPGAs Overview. [Електронний ресурс] / Xilinx Inc. – Режим доступу: http://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf

26. Spartan-6 FPGA Configuration User Guide. Xilinx, 2013. [Електронний ресурс] / Xilinx Inc. – Режим доступу: http://www.xilinx.com/support/documentation/user_guides/ug380.pdf

27. XA Spartan-6 Automotive FPGA Family Overview. Xilinx, 2013. [Електронний ресурс] / Xilinx Inc. – Режим доступу: http://www.xilinx.com/support/documentation/data_sheets/ds170.pdf

28. XA Spartan-3A AutomotiveFPGA Family Data Sheet. Xilinx, 2011. [Електронний ресурс] / Xilinx Inc. – Режим доступу: http://www.xilinx.com/support/documentation/data_sheets/ds681.pdf

29. Aerospace and Defense. Xilinx, 2013. [Електронний ресурс] / Xilinx Inc. – Режим доступу: http://www.xilinx.com/applications/aerospace-and-defense/

30. CoolRunner-II CPLDs. Xilinx, 2013. [Електронний ресурс] / Xilinx Inc. – Режим доступу: http://www.xilinx.com/products/silicon-devices/cpld/coolrunner-ii/index.htm

31. Zynq-7000 All Programmable SoC. Xilinx, 2007. [Електронний ресурс] / Xilinx Inc. – Режим доступу:http://www.xilinx.com/products/silicon-devices/soc/zynq-7000/index.htm

32. Stratix 10 FPGAs and SoCs: Delivering the Unimaginable. Altera. 2013. [Електронний ресурс] / Altera corporation. – Режим доступу: http://www.altera.com/devices/fpga/stratix-fpgas/stratix10/stx10-index.jsp#sthash.es0ZNiQE.dpuf

33. Cyclone V FPGAs: Lowest System Cost and Power. Altera. 2013. [Електронний ресурс] / Altera corporation. – Режим доступу: http://www.altera.com/devices/fpga/ cyclone-v-fpgas/cyv-index.jsp

34. Arria 10 Device Overview. [Електронний ресурс] / Altera corporation. – Режим доступу: http://www.altera.com/literature/hb/arria-10/arria_10_aib.pdf

35. MAX V Device Handbook [Електронний ресурс] / Altera corporation. – Режим доступу: http://www.altera.com/literature/hb/max-v/max5_handbook.pdf

36. Fusion Technology White Paper [Електронний ресурс] / Microsemi Corporation. – Режим доступу: http://www.microsemi.com/index.php?option=com_docman&task=doc_download&gid=131568

37. QuickLogic – Innovative CSSP Solutions for Mobile Device.[Електронний ресурс] / QuickLogicCorporation. – Режим доступу: http://www.quicklogic.com/assets/pdf/corp/QuickLogic-Corporate-Overview.pdf

38. QuickLogic PolarPro Solution Platforms Brief. [Електронний ресурс] / QuickLogicCorporation. – Режим доступу: http://www.quicklogic.com/assets/pdf/solution_platform_briefs/PolarPro_SPB.pdf

39. MachXO PLD Family. [Електронний ресурс] / Laticce semiconductor Corporation. - 2013. – Режим доступу: http://www.latticesemi.com/Products/FPGAandCPLD/MachXO.aspx

40. iCE40 FPGA Family. [Електронний ресурс] / Laticce semiconductor Corporation. - 2013. – Режим доступу: http://www.latticesemi.com/Products/FPGAandCPLD/iCE40.aspx

41. A Generation Ahead for All Programmable & Smarter Systems. [Електронний ресурс] / Xilinx Inc. – Режим доступу: http://www.xilinx.com/aboutus/corporate_overview.pdf

42. Kintex-7 FPGA Family. [Електронний ресурс] / Xilinx Inc. – Режим доступу: http://www.xilinx.com/products/silicon-devices/fpga/kintex-7/index.htm

43. В.Зотов „Особенности архитектуры нового поколения ПЛИС FPGA фирмы Xilinx серии Spartan-6” /Компоненты и технологии №9 2009 стр. 62-70

44. Spartan-3A FPGA Family: Data Sheet. Xilinx, 2008. [Електронний ресурс] / Xilinx Inc. – Режим доступу: http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf

45. Xilinx: MicroBlaze Processor Reference Guide.(v 4.0). 2004[Електронний ресурс] / Xilinx Inc. – Режим доступу: http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_4/mb_ref_guide.pdf

46. The ARM Cortex -A9Processors. [Електронний ресурс] / ARM Holdings plc– White Paper. – 2009 – Режим доступу: http://www.arm.com/files/pdf/ARMCortexA-9Processors.pdf

47. Сергиенко А.М. VHDL для проектирования вычислительных устройств. - К.: -"ДиаСофт". -2003. -210 с.

48. Грушвицкий, Р.И. Проектирование систем на микросхемах программируемой логики / Р.И. Грушвицкий, А.Х. Мурсаев, Е.П. Угрюмов. - СПб.: БХВ-Петербург, 2002. - 608 с.

49. Mike Santarini Xilinx Customer Innovation: 85,000 to 2.5 Billion Transistors and Beyond [Електронний ресурс] / Xilinx Inc. - Xcelljournal. - 2010. - Special Issue – pp. 8-15. - Режим доступу: http://www.xilinx.com/publications/archives/xcell/Xcell-customer-innovation-2010.pdf

50. Keating M. Reuse Methodology Manual for System-On-a-Chip Design / M. Keating, P. Bricaud // Kluwer Academic Publishers, 1999. - 224 p.

51. IEEE, Standard VHDL Language Reference Manual. Standard 1076-1993.- New York, NY: IEEE, 1993.

52. IEEE, Standard Verilog Hardware Description Language Reference Manual. Standard 1364-1995, New York, NY: IEEE, 1995.

53. ModelSim. ASIC and FPGA design. [Електронний ресурс]/ Mentor Graphics Corp.-2013. Режим доступу: http://www.mentor.com/products/fv/modelsim/

54. Active-HDL™. Design Creation and Simulation. [Електронний ресурс]/ Aldec, Inc.- 2013. Режим доступу: http://www.aldec.com/en/products/fpga_simulation/active-hdl

55. Altium Designer. [Електронний ресурс]/ Altium Limited.-2013. Режим доступу: http://www.altium.com/en/products/altium-designer

56. SOCRATES. [Електронний ресурс]/ Duolog Technologies.-2013. Режим доступу: http://www.duolog.com/products/socrates/

57. Комолов Д.А., Мялык Р.А. Системи автоматизированого проектирования фирмы Altera MAX +plus II и Quartus II. Краткое описание и самоучитель. – М: ИП РадиоСофт, 2002. – 352 с.

58. ISE Design Suite 13: Release Notes Guide [Електронний ресурс] / Xilinx Inc. -2011.- Режим доступу: http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_2/irn.pdf

59. Vivado Design Suite. [Електронний ресурс] / Xilinx Inc. – White Paper 2012.- Режим доступу: http://www.xilinx.com/support/documentation/white_papers/wp416-Vivado-Design-Suite.pdf

60. Vivado™ - Новое средство разработки XILINX. [Електронний ресурс] / ЗАО "КТЦ "Инлайн Груп". Режим доступу: http://plis.ru/docum#32/70

61. 4X Faster Implementation [Електронний ресурс] / Xilinx Inc. -2012.- Режим доступу: http://www.xilinx.com/products/design-tools/vivado/prod-advantage/faster-implementation/index.htm

62. IEEE Std. 1666-2005 IEEE Standard SystemC Language Reference Manual, 31

63. Pellerin D. Practical FPGA Programming in C / D. Pellerin, S. Thibault // Pearson Education, Inc., Upper Saddle River, NJ, 2005.

64. Handel-C Language Reference Manual For DK. Version 4. // Celoxica Limited,2005. -348p.

65. Genest G. Programming an FPGA-based Super Computer Using a C-to-VHDL Compiler: DIME-C / G. Genest, R. Chamberlain, R. Bruce // Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference, 5-8 Aug. 2007. - P.280-286.

66. Тарасенко В.П., Михайлюк А.Ю., Тесленко О.К., Осипов О.С. Методологічні та термінологічні аспекти інформаційної стійкості освітніх

комп'ютерних технологій та мереж // Радіоелектронні та комп'ютерні системи.- 2006. - №7, - с. 28-31.

67. Musker D.C. : Reverce engineering. Protecting & Exploiting intellectual property in Electronics, 1998.

68. Device Reliability Report. Xilinx Inc., User Guids, 2015, http://www.xilinx.com/support/documentation/user_guides/ug116.pdf.

69. Jameel Hussein and Gary Swif : Mitigating Single-Event Upsets. Xilinx Inc. WP395 (v1.1) May 19, 2015, http://www.xilinx.com/support/documentation/white_papers/wp395-Mitigating-SEUs.pdf.

70. R. Rajaei, B. Asgari, M. Tabandeh, M. Fazeli: Single Event Multiple Upset-Tolerant SRAM Cell Designs for Nano-scale CMOS Technology. Turkish Journal of Electrical Engineering & Computer Sciences, 2016.

71. M. McLean and J. Moore,:FPGA-Based Single Chip Cryptographic Solution. Military Embedded Systems, 2007.

72. G. Qu,M. Potkonjak: Intellectual Property Protection in VLSI Designs: Theory and Practice. Published by Springer, 2011.

73 . T. Wollinger, J. Guajardo, C. Paar **:** Security on FPGAs: State-of-the-art implementations and attacks. Journal ACM Transactions on Embedded Computing Systems, v.3-i.3, pp. 534-574, 2004,

74. S. McNeil: Solving Today's Design Security Concerns. WP365, 2012, https://www.xilinx.com/support/documentation/white_papers/wp365_Solving_Security_Concerns.pdf.

75. HMAC: Keyed-Hashing for Message Authentication. https://www.ietf.org/rfc/rfc2104.txt .

76. Advanced Encryption Standard (AES). (FIPS PUB 197) , https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf.

77. K.Wilkinson: Using Encryption to Secure a 7 Series FPGA Bitstream. XAPP1239 (v1.0) April 15, 2015, https://www.xilinx.com/support/documentation/application_notes/xapp1239-fpga-bitstream-encryption.pdf.

78. R. Kuramoto: eFUSE Programming on a Device Programmer. https://www.xilinx.com/support/documentation/application_notes/xapp1260-efuse-programmer.pdf.

79.Amir Moradi, Tobias Schneider : Improved SideChannel Analysis Attacks on Xilinx Bitstream Encryption of 5 6 and 7 Series/ Constructive Side-Channel Analysis and Secure Design: 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016.

80. FIPS-198-1, Keyed-Hash Message Authentication Code, Federal Information Processing Standards, U.S. National Institute of Standards and Technology, http://www.nist.gov/itl/upload/FIPS-198-1_final.pdf.

81. Ali Saeed, Muhammad Khalil Shahid: Enhancing Hash Function Selection Techniques Based on Message Contents. IJCNIS, vol.4, no.1, pp.45-51, 2012.

82. E. Gabidulin, N. Pilipchuk: Error and erasure correcting algorithms for rank codes. Des. Codes Cryptogr. (2008).

83.LogiCORE IP Soft Error Mitigation Controller v3.4.1. Product Guide. September 30, 2015, https://www.xilinx.com/support/documentation/ip_documentation/sem/v3_4/pg036_sem.pdf.

84. Eke O. Bartholomew, Ebong A. Oscar: Error Detection in a Multi-user Request System Using Enhanced CRC Algorithm, IJITCS, vol.6, no.9, pp.14-23, 2014

85. M. Tehranipoor, C. Wang "Introduction to Hardware Security and Trust " Springer August 2011.

86. M. Tehranipoor, F. Koushanfar "A Survey of Hardware Trojan Taxonomy and Detection " IEEE Design and Test of Computers pp. 10-25 January-February 2010.

87. M. Banga and M. Hsiao "A novel sustained vector technique for the detection of hardware trojans " IEEE VLSI Design pp. 327 -332 Jan. 2009.

88. M. Potkonjak, "Synthesis of trustable ics using untrusted CAD tools", In Proceedings of the 2010 47th ACM/IEEE Design Automation Conference (DAC), 2010, pp. 633–634.

89. A. Waksman, S. Sethumadhavan,"Silencing hardware backdoors", In Proceedings of the 2011 IEEE Symposium on Security and Privacy (SP ’11), Berkeley/Oakland, CA, USA, 2011, pp. 49–63.
90. M. Beaumont, B. Hopkins, T. Newby, "Safer path: Security architecture using fragmented execution and replication for protection against Trojaned hardware", In Proceedings of the IEEE Conference on Design, Automation and Test in Europe (DATE ’12), Dresden, Germany, 2012, pp. 1000–1005.
91. A. Baumgarten, A. Tyagi, J. Zambreno, "Preventing IC piracy using reconfigurable logic barriers", IEEE Design & Test of Computers (2010), 27(1), pp. 66–75.
92. M. Tehranipoor, F. Koushanfar, "A survey of hardware Trojan taxonomy and detection", IEEE Design & Test of Computers (2010), 27(1), pp. 10–25.
93. X. Wang, H. Salmani, M. Tehranipoor, J. Plusquellic, "Hardware Trojan detection and isolation using current integration and localized current analysis", In Proceedings of the IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems (DFT ’08), Washington, DC, USA, 2008, pp. 87–95
94. R. Rad, J. Plusquellic, M. Tehranipoor, "A sensitivity analysis of power signal methods for detecting hardware Trojans under real process and environmental conditions", IEEE Transactions on Very Large Scale Integrated Systems (VLSI) (2010), 18(12), pp. 1735–1744.
95. Y. Jin, Y. Makris, "Hardware Trojan detection using path delay fingerprint", In Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust (HOST ’09), Washington, DC, USA, 2008, pp. 51–57.
96. M. Banga, M. Chandrasekar, L. Fang, M. Hsiao, "Guided test generation for isolation and detection of embedded Trojans in Ics", In Proceedings of the 18th ACM Great Lakes symposium on VLSI (GLSVLSI ’08), New York, NY, USA, 2008, pp. 363–366.

97. H. Salmani, M. Tehranipoor, J. Plusquellic, "New design strategy for improving hardware Trojan detection and reducing Trojan activation time", In Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust (HOST '09), Washington, DC, USA, 2009, pp. 66–73

98. M. Banga, M. S. Hsiao, "Vitamin: Voltage inversion technique to ascertain malicious insertions in Ics", In Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust (HOST '09), Washington, DC, USA, 2009, pp. 104–107.

99. J. Rajendran, V. Jyothi, O. Sinanoglu, R. Karri, "Design and analysis of ring oscillator based design-for-trust technique", In Proceedings of the IEEE 29th VLSI Test Symposium (VTS), Dana Point, CA, USA, 2011, pp. 105–110.

100. Amr Al-Anwar, Mona A. Aboelnaga, Yousra Alkabani, M. Watheq El-Kharashi, Hassan Bedour: Dynamic FPGA Detection and Protection of Hardware Trojan: A Comparative Analysis.

101. A. Al-Anwar, Y. Alkabani, M. W. El-Kharashi, and H. Bedour, ―Hardware Trojan detection methodology for FPGA,‖ in Proceedings of the 2013 IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing (PacRim), Victoria, BC, Canada, pp. 177–182.

102. T. Güneysu, B. Möller, and C. Paar, "Dynamic intellectual property protection for reconfigurable devices," in *Proc. Int. Conf. Field-Program. Technol. (ICFPT)*, Dec. 2007, pp. 169–176.

103. OpenCores. [Електронний ресурс] /OpenCores / Режим доступу: http://opencores.org.

104. National Institute of Standards and Technology (NIST).Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography. NIST Special Publication SP 800-56A, 2006.

105. Altera Corp. Stratix II GX and Stratix III FPGAs, 2006. www.altera.com/products/devices/

106. Xilinx Corp. Virtex2, Virtex 4 and Virtex 5 FPGAs, 2006. www.xilinx.com/products/silicon solutions/

107. National Institute of Standards and Technology (NIST). Recommendation for block cipher modes of operation – the CMAC mode for authentication. NIST Special Publication SP 800-38B, 2005.

108. L. Bossuet, G. Gogniat, and W. Burleson. Dynamically configurable security for SRAM FPGA bitstreams. International Journal of Embedded Systems, 2(12):73–85, 2006.

109. O. K¨ommerling and M. G. Kuhn. Design principles for tamper-resistant smartcard processors. In Proceedings of the USENIX Workshop on Smartcard Technology, 1999.

110. T. Wollinger and C. Paar. How secure are FPGAs in cryptographic applications, 2003.

112. T. Kean. Secure configuration of field programmable gate arrays. In In proceeding of 11th International Conference on Field-Programmable Logic and Applications, FPL2001, Belfast, United Kingdom, 2001.

113. C¸ . K. Koc¸. RSA hardware implementation. Technical report TR801, RSA Data Security, Inc., Aug. 1995.

114. A. Mazzeo, L. Romano, G. PaoloSaggese, and N. Mazzocca. FPGA-based implementation of a serial RSA processor. Design, Automation and Test in Europe Conference and Exposition, pp. 10582–10589, 2003.

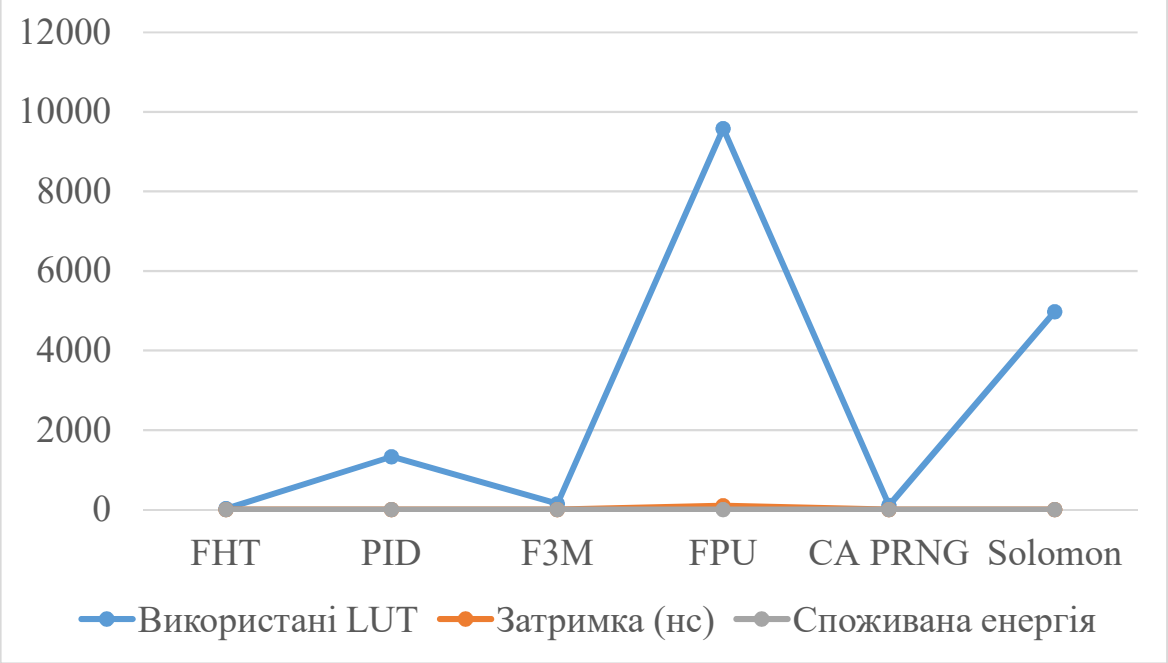
115. Berkeley University. Bee2 project. http://bee2.eecs.berkeley.edu

116. D. Lim. Extracting secret keys from integrated circuits. Master's thesis, 2004.

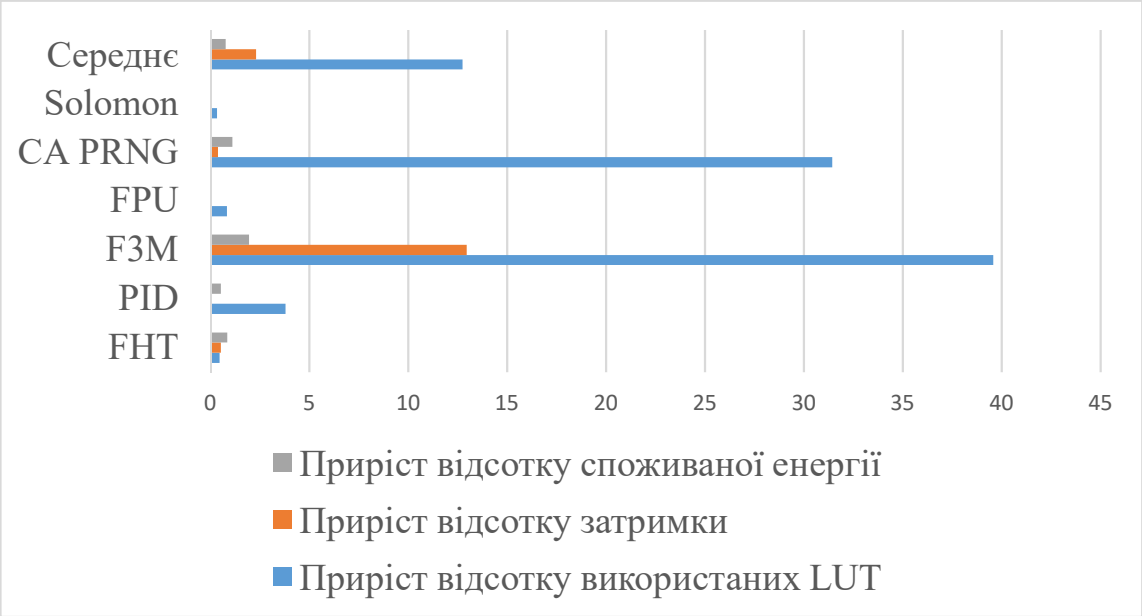
ДОДАТКИ

Додаток 1. Копії графічних матеріалів

Результати синтезу

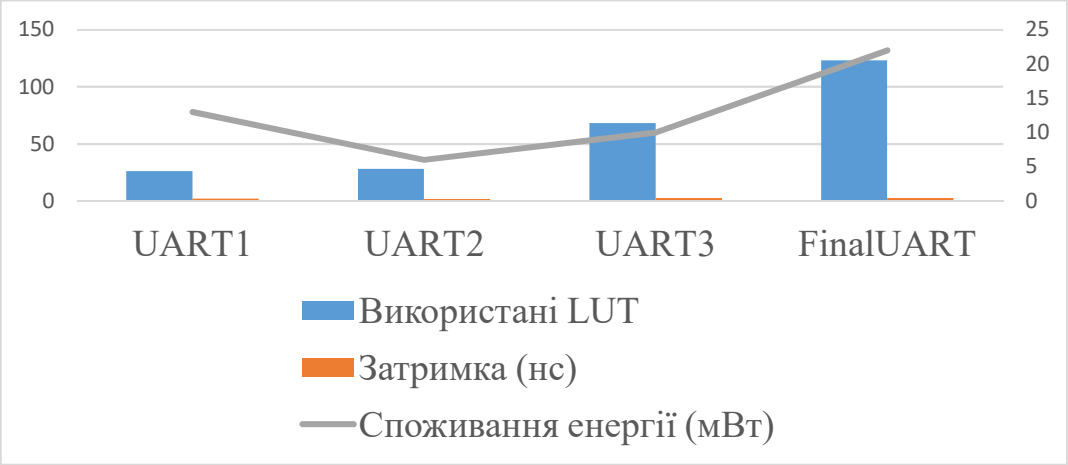


Результати синтезу після проведення обфускації

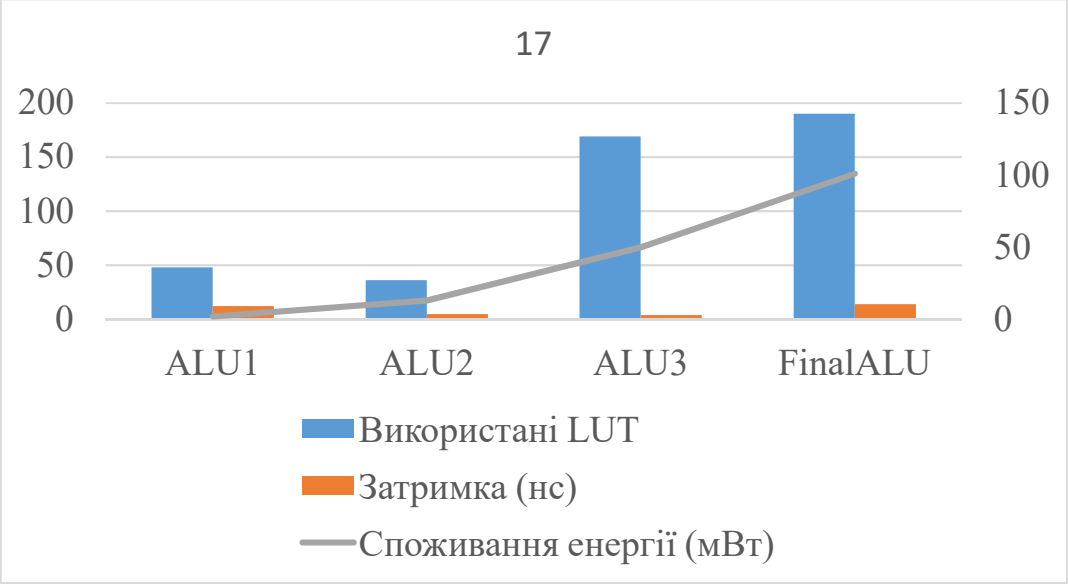


Тарасенко Г.О.

Реалізація методу MV на ядрах UART

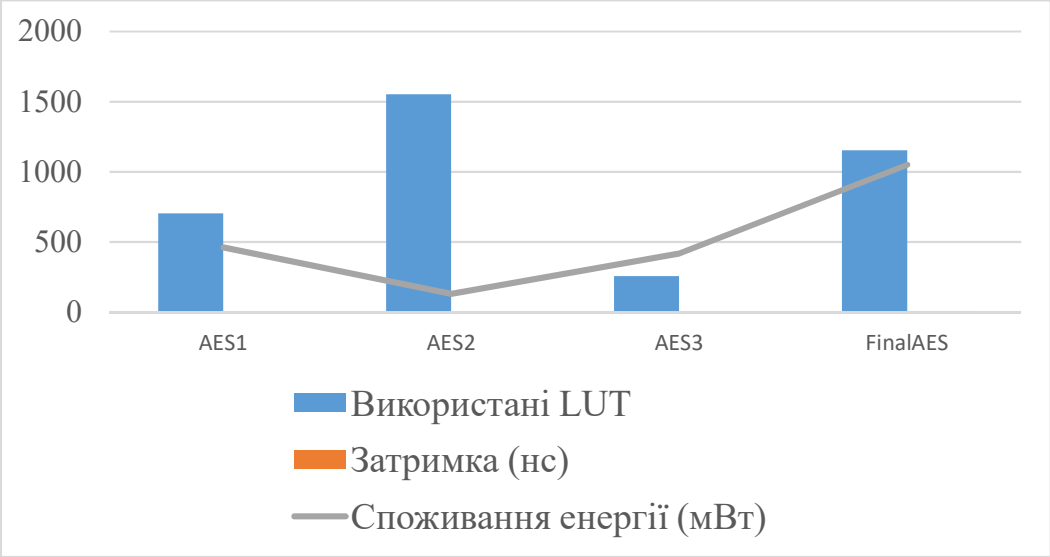


Реалізація методу MCRC на ядрах ALU



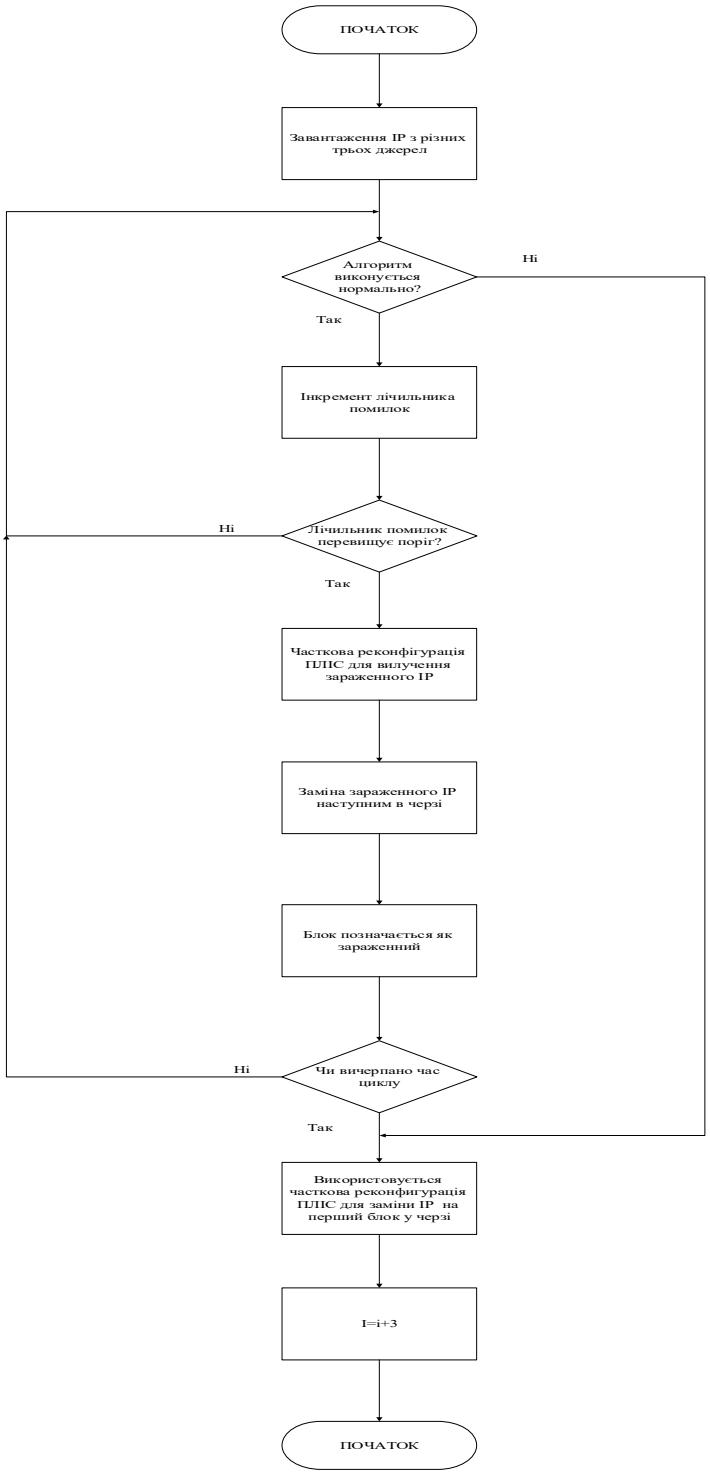
Тарасенко Г.О.

Реалізація методу MVR на ядрах AES



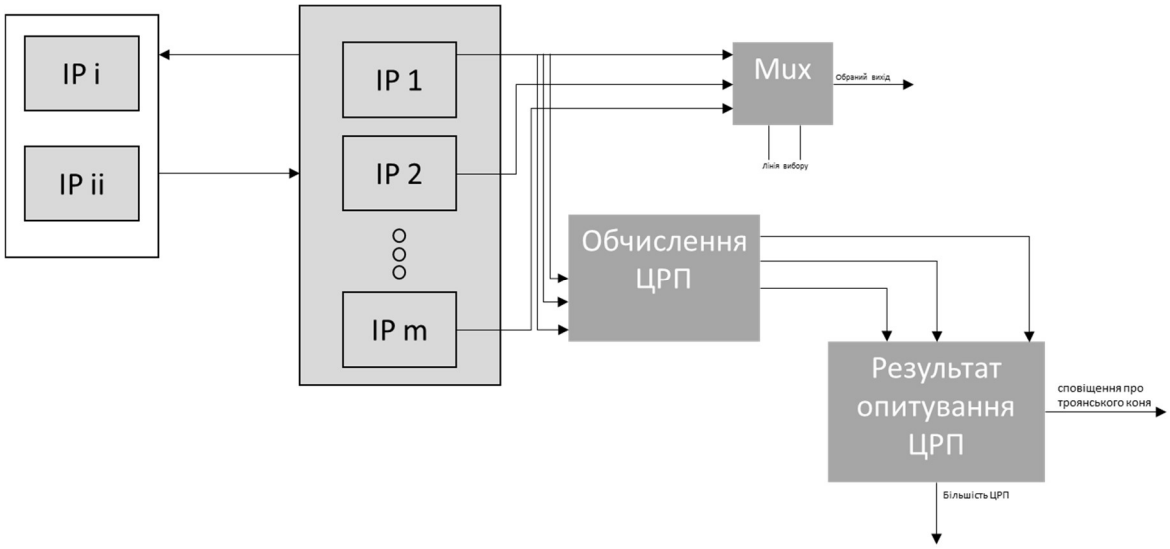
Тарасенко Г.О

Блок-схема алгоритму мультиплексування даних із виходів реконфігурованих IP та реалізація схеми виявлення троянців .



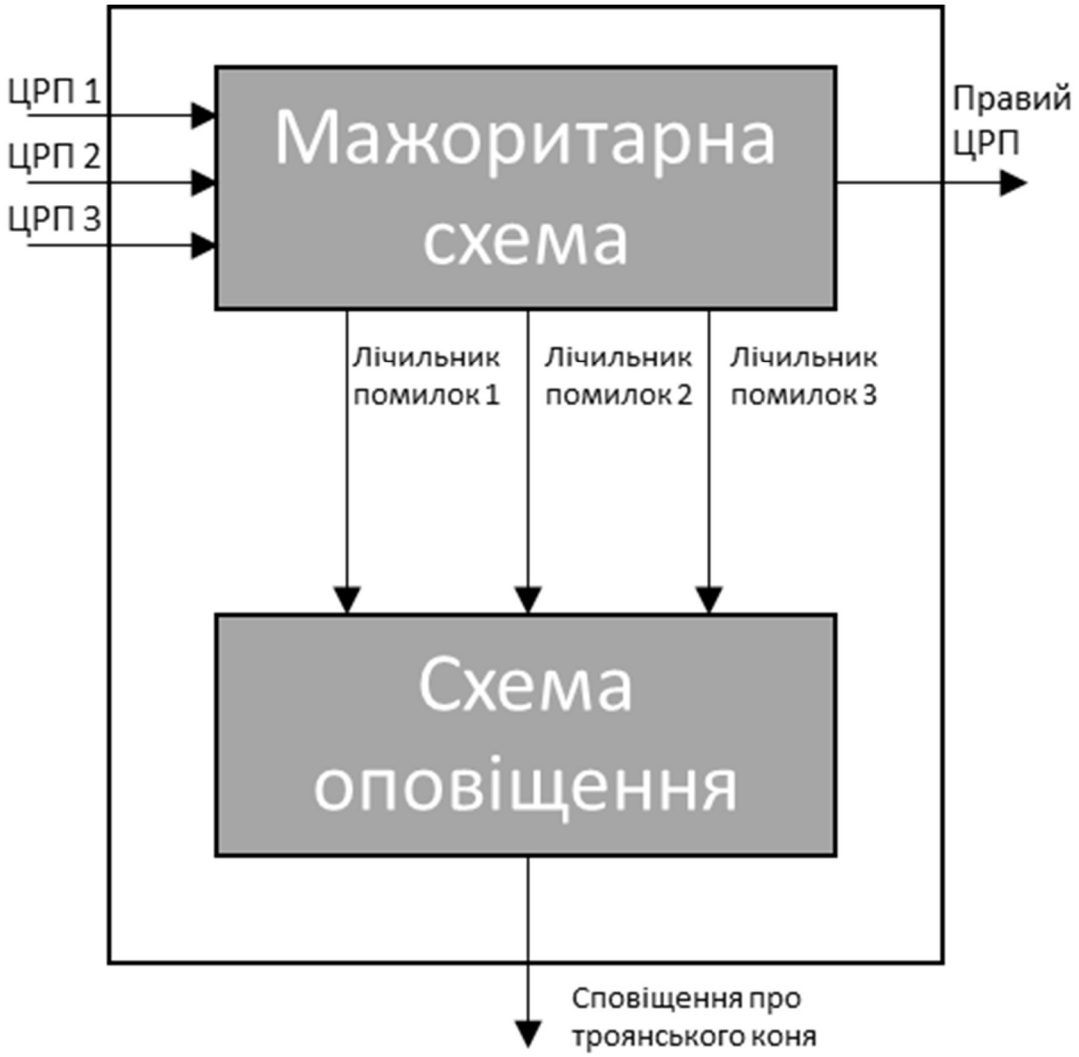
Тарасенко Г.О.

Мультиплексування виходів IP зі змінюваною конфігурацією для вибору вірного і додавання частини CRC для перевірки виходу.



Тарасенко Г.О.

Схема голосування CRC



Тарасенко Г.О.

Додаток 2. Копії публікацій.

УДК 004.932.72

-

Yaroslav M. Klyatchenko, Associate professor, PhD; student Georgii Tarasenko

National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"

SPECIALIZED PROTECTION OF FPGA-BASED COMPUTER DEVICES FROM DIFFERENT THREATS

Abstract.

Yaroslav Klyatchenko, assoc. prof., PhD; Georgii Tarasenko, student
Intellectual property cores protection in specialized computer facilities on FPGA

Recently, new facilities of computer technology are implemented on the basis of programmable logical devices. The increase in the complexity of digital computing devices, especially in the specialized systems of critical analysis translates into the focus of computation means' producers and companies which produce FPGA to the occurrence of situations caused by violations of the proper functioning of digital structures, due to external influences and interventions. The modern level of specialized computing has been achieved through the widespread use of FPGAs and intellectual property cores (IP). Implementing IP protection is a complex and important task.

Introduction

The importance of ensuring the information security of computer devices implemented in FPGA in the context of counteraction to negative phenomena such as unauthorized access to or theft of intellectual property is proven by the presence of threats and weaknesses in such hardware. One of the modern tendencies in the creation of specialized computer means is using FPGA platforms for the development of such digital computing as, for example, SoC (System on a Chip), a system on a programmable chip using FPGA - SoPC (System on a Programmable Chip) and others. In the design of modern SoC are widely used virtual blocks or IP-core (the intellectual property core, IP) implemented in the functionality of the devices, and which provide the necessary algorithms for processing. IP is a complex, tested, verified and optimized functional module, which can be used several times as a key component in the design. As you can see from the practice, using IP allows you to reduce the time until the appearance of the digital product on the market. It simplifies the process of creating computer tools by creating a new interface for communication with IP, minimizing development patterns due to the inclusion of already-released modules, reducing

the time for the verification of design. The current IP market is constantly expanding and developing [1]. But there is a reverse side – what is being created for months or years can be stolen in seconds. Theft of blocks of intellectual property (IP) is possible when cloning (copying) a device or by reverse engineering [2-6]. If in the first case the device is created as close as possible to the original, which cannot always be considered theft, then in the second case the attacker gets to the very essence of the intellectual product - the algorithms of the functioning of the device and the construction of a new specification of the device for further manipulation of this information.

Problem definition

The threats and attacks that accompany the process of implementing computing devices on the FPGA, and contribute to the emergence, first of all, of problems with the use of intellectual property cores, the protection of which from an unauthorized acquaintance, use, forgery, modification, etc. is an urgent task of ensuring information safety of these means.

Algorithm description

There are classical approaches to implementing IP protection. These include protection through patents, copyright, legal mechanisms and the introduction of commercial secrets and instructions at the enterprise. For example, in some countries reverse engineering is allowed only for educational or research purposes. Other classic approach to protection is using design encryption, when, bundled with intellectual property core, the key for decryption is sent to a client, but, again, an agreement on the use of the IP and the key should be made.

FPGA's leading manufacturers offer a wide range of solutions for protecting intellectual property rights - from the introduction of the device identifier and encryption of the configuration sequence to the use of the HMAC, for bitstream authentication, and usage specialized security tools.

For protection against spoofing and Trojans, facilities that use cryptographic authentication demonstrate their effectiveness. To load bitstream encrypted by AES algorithm, an embedded FPGA block is used to transmit bitstream with HMAC authentication method [18]. By using HMAC integrity check mechanism, a message sender produces the message authentication code (MAC) using the secret key and the message. The receiving side (in this context, in the software environment, the hardware side) calculates this hash code for the received message using the same key and compares the results. Both components generate a 256-bit MAC based on a key and a secure hash algorithm (SHA256). If these two values match, then the message is considered verified. So, without knowing the keys for AES and HMAC bitstream, IP cannot be downloaded, modified, or

cloned. If the AES algorithm provides IP protection against copying or reverse engineering, then using the HMAC hashing mechanism guarantees that the bitstream configuration of the device loaded into the FPGA has not been changed. This way, any change in the configuration thread, including a change in the value of at least one bit, is detected.

For almost complete elimination of threats to IP, additional security methods are often used, combined with the security methods described above. It may be used SEU resistant to feedback circuit disconnecting connection with triple redundancy that blocks reading configuration memory via the exchange and techniques directed against reverse engineering.

SEU can affect the cell's memory and cause a change in the value of one or more bits and it is important here where the memory is located. Affected bits may belong to the configuration memory that contains design. This memory is the largest in volume and physically distributed across parts of the device, but only a part of the bits are important for the proper operation of any particular design that is downloaded to the device. Other high-capacity memory elements used to store design state are called block memory. This memory is second to the capacity. The blocks of memory are grouped together and located throughout the FPGA. Distributed memory includes memory elements for storing the status of the design on the FPGA. This type of memory is implemented on the basis of the Configurable Logic Block (CLB) and distributed over the software crystal. Distributed memory is the third memory capacity. The last kind of memory is flip-flops, has a low capacity and is used to store design state. This type of memory is present in all CLBs and is located throughout FPGA. Flip-flops has the highest performance and is the scarcest programmable resource of FPGA.

To counter SEUs, as well as side-channel attack and tamper, special means that are used in the FPGA, realizing background reading bitstream. At the same time, the error detection and correction mechanism, known as the error-correcting code (ECC), is used. This approach simplifies the detection of changes in the configuration memory that may occur either as a result of the SEU or through an attack on the design. Consider the peculiarities of the functioning of such a mechanism. These means of FPGA single crashes mitigation, do not prevent errors, but allow either mitigate effects or fix errors.

Let's evaluate the characteristic features of these means. Implementation of Additional means for intellectual property core protection

The latest programmable logic devices have the ability to erase the encryption key from the corresponding memory cells or contents of configuration and flip-flop memory of FPGA by a special signal. Such a mechanism can be activated in response to an attacker's actions.

In order to prevent cloning of the device, the latest generation FPGAs contain a built-in unique device identifier. This unique identifier (similar to the serial number) is stored in the nonvolatile memory and puts it there at the last

stage of the crystal programming. In a device design, the user can use this unique identifier to implement its own means for protecting IP from tampering.

Conclusions

The modern means that use to interfere with intellectual property core, the diversity of their arsenal of tampering techniques, and the threat to the proper functioning of computing devices FPGAs by SEU, prompts developers of these IP, users and companies of the software makers of the software to develop and implement methods and remedies. These facilities help customers create computing environments on FPGAs that are not only secure but also resistant to cloning and tampering. It has been shown that the application of encryption and cryptographic authentication can significantly increase the level of FPGA design protection, and in combination with the use of eFUSE can create an insurmountable boundary for intruders. Consequences of single failures can be avoided if you use the available hardware to detect and correct errors that is embedded in the FPGA.

References

1. McGrath, D.: Report: Semiconductor IP market to double in five years. EETIMES, 2012, http://www.eetimes.com/document.asp?doc_id=1261490.
2. Wollinger, T., Guajardo, J., Paar, C.**: Security on FPGAs: State-of-the-art implementations and attacks**. Journal ACM Transactions on Embedded Computing Systems, v.3-i.3, pp. 534-574, 2004,
3. Musker, D.C. : Reverce engineering. Protecting & Exploiting intellectual property in Electronics, 1998.
4. Wollinger, T., Paar, C.: How secure are FPGAs in cryptographic applications. LNCS, 2003.
5. Trimberger, S.: Trusted design in FPGAs. Proc. 44 Annual Design Automation Conf., N.Y.: ACM, 2007.
6. Device Reliability Report. Xilinx Inc., User Guids, 2015, http://www.xilinx.com/support/documentation/user_guides/ug116.pdf.
7. McLean, M. and Moore, J.: FPGA-Based Single Chip Cryptographic Solution. Military Embedded Systems, 2007.

Optimization of processor devices based on the maximum indicators of self-correction

Klyatchenko Y.[0000-0003-4236-4059], Tarasenko G., Tarasenko-Klyatchenko O., Tarasenko V., Teslenko O.

National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute",37, Prosp.Peremohy, Kyiv, Ukraine, 03056.
k_yaroslav@scs.ntu-kpi.kiev.ua

Abstract. It is proposed to characterize the state of digital processing devices' outputs with a system of probabilities, where for any binary processing results the probability of getting the same real result is set, if it is theoretically possible. Since the outputs of some devices in their compositions are the inputs of other devices, than such a probability system is universal, that allows taking into account faults of hardware means. For the original inputs the models of probabilistic information converters (data distortion on the initial inputs system) are used, on the outputs of which the indicated probabilities system exists, allowing for distortion of the input data only. Heuristic algorithm for optimal encoding (as per the maximal possible self-correction) of digital machines (as the models of processing devices) is proposed for their structural synthesis. The obtained results enable reaching optimize results on the design stage, that are directed to increase veracity of computer systems processor devices' functioning.

Keywords: Self-correction, probability of correct operation, Boolean functions.

1 Introduction

In the process of operation of control and monitoring systems of the different objects, especially of the crucial use objects, the substantial indicator of their efficiency is the veracity of controlling influences and obtained data. In the computer engineering it is commonly believed [1], that with input data distortion or with faults the veracity of result is not guaranteed. At the same time, computer devices can contain latent properties for correction of input data distortions and failures – the phenomenon of self-correction (The authors accentuate the term "self-correction" and deliberately avoid the term "auto-correction", since the latter has an associate connection to the term "automatically", the property mentioned above is determined by peculiar features of the functions implemented without any automatism). Probability increase of the correct devices operation' in conditions of the input data distortions and failures with the aide of self-correction, determines the absolute value of self-correction effect [2]. Ensuring maximal possible values of self-correction effect on the design stage allows increasing the control quality and objects monitoring. To carry out the according researches and obtained results comparison it is required to make values calculation for self-correction effect. According to [3-7] in the computer engineering it is common that a necessity appears to implement multiple-output combinational schemes upon fractionally-determined Boolean functions, particularly at the stage of encoding of the alphabets' letters of inputs, states and outputs, upon the structural synthesis of digital machines. The goal of the work is to create methods to reduce the number of calculations of self-correction effect' values and the method for optimization of project solutions based on self-correction parameter to use them at the stage of computer systems development.

2 Self-correction indices calculation

2.1 Actual implementations usage

One of the directions to reducing the volume of calculations of self-correction indices involves using the fact that multiple-digits processing devices are implemented using devices of less-digits capacity. In the work [3] the data distortion' probabilities system is proposed, that is adequate both for input data and results. That has become the basement for the technique creation that enables substantial reduction of computational complexity for self-correction properties' estimates under the condition of corresponding random values, for example upon the following distributing decomposition of the Boolean functions: $f(x_1,x_2,\ldots,x_n)=f_1(x_1,x_2,\ldots,x_t,f_2(x_{t+1},x_{t+2},\ldots,x_n))$.

In general case the distributing decompistion is determined the floowing way :

$$f(x_1,x_2,\ldots,x_n)=f_1(x_1,x_2,\ldots,x_t,f_{21}(x_{t+1},x_{t+2},\ldots,x_n),\ldots,f_{2r}(x_{t+1},x_{t+2},\ldots,x_n)). \quad (1)$$

The values of $f_{21}(x_{t+1},x_{t+2},\ldots,x_n),\ldots,f_{2r}(x_{t+1},x_{t+2},\ldots,x_n)$ function are not independent, that does not allow using the indicated technique immediately to calculate the self-correction properties of the f_1 function. The task is to create more universal technique that would allow possibility to calculate the self-correction indicators for variables, separate functions and its corteges.

We will call the cortege of the F functions a normalized set of functions, that depend on the same variables: $F(x_{t+1},x_{t+2},\ldots,x_n)=<f_{21}(x_{t+1},x_{t+2},\ldots,x_n),\ldots, f_{2r}(x_{t+1},x_{t+2},\ldots,x_n)>$. Since the cortege functions as random values are not mutually independent, then the probability of the function' cortege value in general cannot be determined using the product of separate functions' probabilities.

For the probability of the functions' cortege it is proposed to use two-dimensional array JF, where the span of each dimension does not exceed 2^r. Assume e as a value of the function' cortege, obtained in the result of data distortions, w is a correct (faultless) value, $(e,w\in\{0,1,\ldots, 2^r-1\})$. Then the element of $JF[e,w]$ array is a sum of input data probabilities, when in result of data distortion effect the functions' cortege value F will be equal e, and not w. In case when $e=w$ the probability of self-correction will be located in the corresponding array element.

The proposed array of probabilities is easily determined for a separate independent variable. Indeed, accordingly [3] we have $p^0_{gi} + p^1_{gi} = (g_{0i}p^0_i + g_{ci}{}^0p^0_i + g_{ei}{}^0p^1_i) + +(g_{0i}p^1_i + g_{ci}{}^1p^1_i + g_{ei}{}^1p^0_i)$, where p^0_{gi} , p^1_{gi} are probabilities of 0 and 1 values on the i-input in the result of distortions; g_{0i} is the probability of any distortions absence; $g_{ci}{}^0$ is the sum of the determined distortions, upon which the null-value x_i does not change (incoming self-correction); $g_{ei}{}^0$ is the sum of determined distortions probability upon which the singular x_i value change to the null-value; $g_{ci}{}^1$ is the sum of probabilities, upon which the singular x_i value does not change (incoming self-correction); $g_{ei}{}^1$ is the sum of determined distortions probability, upon which the x_i null value change to the singular. Then $Jx_i[0,0] = g_{0i}p^0_i + g_{ci}{}^0p^0$, $Jx_i[0,1] = g_{ei}{}^0p^1_i$, $Jx_i[1,0] = g_{ei}{}^1p^0_i$, $Jx_i[1,1]= g_{0i}p^1_i + g_{ci}{}^1p^1_i$.

The proposed technique for probabilities calculation is more general, in comparison with technique proposed in [3], since it has the sense for the functions' cortege, for the input independent data and for the separate Boolean function. For instance, for the

function $f(x_1,x_2,\ldots,x_n)$ we have: *Jf[0,0]* is probability of correct 0-value, *Jf[1,1]* is probability of correct 1-value,*Jf[0,1]* is probability of false 0-value,*Jf[1,0]* is probability of false 1-value.

Let's examine the self-correction indicators for the $f(x_1,x_2,\ldots,x_n)$ function under conditions of determined incoming distortions regardless of the superposition (1). As in given in [3], we will determine the value of $\boldsymbol{M(f)}$ of all the corteges (organized variables values' sets) $A_j = \langle a_{1j},a_{2j},\ldots,a_{nj}\rangle$, $(j=0,1,\ldots,2^n\text{-}1)$. Assume $C_l = \langle c_{1l},c_{2l},\ldots,c_{nl}\rangle$ $(l=0,1,\ldots,2^n\text{-}1)$ is the cortege of variables values, that consists of the A_j cortege as the result of distortions effect. The probability $P[C_j,A_j]$ of the A_j cortege to the C_l cortege is determined as follows:

$$P[C_l, A_j] = \prod_{i=1}^{n} Jx_i[c_{il}, a_{ij}] . \tag{2}$$

Let us make up a set of $\boldsymbol{H_{(1,n)}}$ products. Since any A_j cortege can transform to any cortege from $\boldsymbol{M(f)}$ as the result of distortions, then the quantity of $\boldsymbol{H_{(1,n)}}$ equals. Let us divide the $\boldsymbol{H_{(1,n)}}$ set into the following subsets (classes):

- Subset $\boldsymbol{K^0_w(f)}$, that contains products where $C_l = A_j$, a $f(A_j) = 0$;
- Subset $\boldsymbol{K^1_w(f)}$, that contains products where $C_l = A_j$, a $f(A_j) = 1$;
- Subset $\boldsymbol{K^0_c(f)}$, that contains products where $C_l \neq A_j$, a $f(C_l) = f(A_j) = 0$;
- Subset $\boldsymbol{K^1_c(f)}$, that contains products where $C_l \neq A_j$, a $f(C_l) = f(A_j) = 1$;
- Subset $\boldsymbol{K^0_e(f)}$, that contains products where $C_l \neq A_j$, a $f(C_l) = 0, f(A_j)=1$;
- Subset $\boldsymbol{K^1_e(f)}$, that contains products where $C_l \neq A_j$, a $f(C_l) = 1, f(A_j) = 0$.

Adding the products' values of the $\boldsymbol{K^0_w(f)}$ and $\boldsymbol{K^0_c(f)}$ we shall determine the value of *Jf[0,0]* as the possibility of the correct 0 value. Adding the products' values of the $\boldsymbol{K^1_w(f)}$ and $\boldsymbol{K^1_c(f)}$ we shall determine the value of *Jf[1,1]* as the possibility of the correct 1 value. Adding the products' values of the $\boldsymbol{K^0_e(f)}$ we shall determine the value of *Jf[0,1]* as the possibility of the false null-value of the *f* function. Adding the products' values of the $\boldsymbol{K^1_e(f)}$ we shall determine the value of *Jf[0,1]* as the possibility of the false single-value of the *f* function. Notice that self-correction possibility is determined by the products' sum of $\boldsymbol{K^0_c(f)}$ and $\boldsymbol{K^1_c(f)}$ classes.

Further we examine GF-array formation for the functions cortege of $F(x_{t+1},x_{t+2},\ldots,x_n)$. Let's mark $\boldsymbol{M(F)}$ as a set of values corteges of $x_{t+1},x_{t+2},\ldots,x_n$ arguments, $Bs = \langle b(t+1),s,b(t+2),s,\ldots,bn,s\rangle$ from $\boldsymbol{M}(F, (s=0,1,\ldots,2^{n-t}\text{-}1)$. $D_u = \langle d_{(t+1),u},d_{(t+2),u},\ldots,d_{n,u}\rangle$ $(u=0,1,\ldots,2^{n-t}\text{-}1)$, cortege from $\boldsymbol{M(F)}$, that is formed from the B_s in the result of distortions effect. The probability of $P[D_u,B_s]$ distortion of B_s cortege to the D_u cortege is determined as follows:

$$P[D_u, B_s] = \prod_{i=t+1}^{n} Jx_i[d_{iu}, b_{is}] . \tag{3}$$

Let's generate a $\boldsymbol{H_{(t+1,n)}}$ products set (3). Assume *e* and *w* any possible values of the *F* functions cortege. Let's decompose the set $\boldsymbol{H_{(t+1,n)}}$ to subset $\boldsymbol{K(e,w)}$ each contains all products, for which $e=F(D_u)$, $w= F(B_s)$ is valid. Then *JF[e,w]* is element of the *JF* array will contain the sum of all products of the $\boldsymbol{K(e,w)}$ subset.

Let's examine the formation of Jf_1 array for the $f_1(x_1,x_2,\ldots,x_t,F)$ function. Let's mark $A_{\alpha,\gamma} = \langle a_{1\alpha},a_{2\alpha},\ldots,a_{t\alpha},w_\gamma\rangle$, $(\alpha=0,1,\ldots,2^t\text{-}1,\ \gamma=0,1,\ldots,2^r\text{-}1$ as the cortege of variables of $x_1,x_2,\ldots,x_t$ values and values of the *F* functions' cortege, $C_{\beta,\delta} = \langle c_{1\beta},c_{2\beta},\ldots,c_{t\beta},\ e_\delta\rangle$ $(\beta=0,1,\ldots,2^t\text{-}1,\ \delta=\gamma=0,1,\ldots,2^r\text{-}1)$ are the values' cortege and the values of functions' cortege *F* that is formed from $A_{\alpha,\gamma}$ as the result of distortions effect. The probability $P[C_{\beta,\delta},\ A_{\alpha,\gamma}]$ of cortege $A_{\alpha,\gamma}$ distortion to $C_{\beta,\delta}$ cortege is determined as follows

$$P[C_{\beta,\delta}, A_{\alpha,\gamma}] = (\prod_{i=1}^{t} Jx_i[c_{i\beta}, a_{i\alpha}])(JF[e_\delta w_\gamma]) . \tag{4}$$

Let's generate the $\boldsymbol{H_{(1,t+r)}}$ set of products (4). The quantity of elements of $\boldsymbol{H_{(1,t+r)}}$ equals $4^t x 2^{r+1}$. Let's decompose the $\boldsymbol{H_{(1,t+r)}}$ set into the following subsets (classes):

- Subset $\boldsymbol{K^0_w(f)}$, that contains products where $C_{\beta,\delta} = A_{\alpha,\gamma}$, a $f_1(A_{\alpha,\gamma}) = 0$;
- Subset $\boldsymbol{K^1_w(f_1)}$, that contains products where $C_{\beta,\delta} = A_{\alpha,\gamma}$, a $f_1(A_{\alpha,\gamma}) = 1$;
- Subset $\boldsymbol{K^0_c(f_1)}$, that contains products where $C_{\beta,\delta} \neq A_{\alpha,\gamma}$, a $f_1(C_{\beta,\delta}) = f_1(A_{\alpha,\gamma}) = 0$;
- Subset $\boldsymbol{K^1_c(f_1)}$, that contains products where $C_{\beta,\delta} \neq A_{\alpha,\gamma}$, a $f_1(C_{\beta,\delta}) = f_1(A_{\alpha,\gamma}) = 1$;

- Subset $\boldsymbol{K^0_e(f_1)}$, that contains products where $C_{\beta,\delta} \neq A_{\alpha,\gamma}$, a $f_1(C_{\beta,\delta})=0, f_1(A_{\alpha,\gamma}) = 1;$
- Subset $\boldsymbol{K^1_e(f_1)}$, that contains products where $C_{\beta,\delta} \neq A_{\alpha,\gamma}$, a $f_1(C_{\beta,\delta})=1, f_1(A_{\alpha,\gamma})=0.$

By the adding products' values of the $\boldsymbol{K^0_w(f_1)}$ and $\boldsymbol{K^0_c(f_1)}$ classes we shall determine $Jf_1[0,0]$ value as the probability of correct value of 0. Adding the products' values of the $\boldsymbol{K^1_w(f_1)}$ and $\boldsymbol{K^1_c(f_1)}$ we shall determine the value of $Jf_1[1,1]$ as the possibility of the correct 1 value. Adding the products' values of the $\boldsymbol{K^0_e(f_1)}$ we shall determine the value of $Jf_1[1,0]$ as the possibility of the false null-value of the f_1 function. Adding the products' values of the $\boldsymbol{K^1_e(f)}$ we shall determine the value of $Jf_1[1,0]$ as the possibility of the false 1-value of the f_1 function. Notice that self-correction possibility is determined by the products' sum of $\boldsymbol{K^0_c(f_1)}$ and $\boldsymbol{K^1_c(f_1)}$ classes.

Assertion. The Jf and Jf_1 arrays are identical.

Proving. Assume $\hat{A}=<a_1,a_2,...,a_t>$ as a random cortege of variables' values $x_1,x_2,...,x_t$, $\hat{C} = <c_1,c_2,...,c_t>$ is some distortion of the $\hat{A}$ cortege. $P[\hat{C}, \hat{A}] = Jx_1[c_1, a_1] \times Jx_2[c_2, a_2] \times ... \times Jx_t[c_t, a_t]$ is the probability of such distortion. According to the distribution rule in the field of real numbers, by the grouping of corresponding products (2) from the set $\boldsymbol{H_{(1,n)}}$, we shall create subset of $\boldsymbol{Q}(\hat{C}, \hat{A}\ e,w) = P[\hat{C}, \hat{A}] \times \boldsymbol{K}(e,w)$. When forming classes of $\boldsymbol{H_{(1,n)}}$ set any products from the $\boldsymbol{Q}(\hat{C}, \hat{A}\ e,w)$ set can be used independently. At the same time, when forming classes of $\boldsymbol{H_{(1,t+r)}}$ set one product is used ($P[\hat{C}, \hat{A}] \times JF[e,w]$). Consequently, to prove Jf and Jf_1 identical it is sufficient to ascertain that upon any $\hat{C}$, $\hat{A}$, e and w one of the following correlations are preserved – $\boldsymbol{Q}(\hat{C}, \hat{A},e,w) \subset \boldsymbol{K^0_w(f)} \cup \boldsymbol{K^0_c(f)}$, or $\boldsymbol{Q}(\hat{C}, \hat{A},e,w) \subset \boldsymbol{K^1_w(f)} \cup \boldsymbol{K^1_c(f)}$, or $\boldsymbol{Q}(\hat{C}, \hat{A},e,w) \subset \boldsymbol{K^0_e(f)}$, or $\boldsymbol{Q}(\hat{C}, \hat{A},e,w) \subset \boldsymbol{K^0_e(f)}$.

Let's mark $B_0 = <b_{(t+1),0}\ ,\ b_{(t+2),0},...,b_{n,0}>$, $B_1 = <b_{(t+1),1}\ ,\ b_{(t+2),1},...,b_{n,1}>$ are random corteges from $\boldsymbol{M(F)}$, $D_0 = <d_{(t+1),0}\ ,\ d_{(t+2),0},...,d_{n,0}>$ and $D_1 = <d_{(t+1),1}$ and $D_1 = <d_{(t+1),1},\ d_{(t+2),1},...,d_{n,1}>$ are distorted corteges B_0 and B_1 accordingly. Using concatenation we shall create the $A_0 = \hat{A} \,||\, B_0$, $A_1 = \hat{A} \,||\, B_1$, $C_0 = \hat{C} \,||\, D_0$, $C_1 = \hat{C} \,||\, D_1$ corteges. Let's determine probability of corteges' transformation:

$$P[D_0, B_0] = Jx_{t+1}[d_{(t+1),0},b_{(t+1),0}] \times ... \times Jx_n[d_{n0}, b_{n0}] \ . \tag{5}$$

$$P[D_1, B_1] = Jx_{t+1}[d_{(t+1),1},b_{(t+1),1}] \times ... \times Jx_n[d_{n1}, b_{n1}] \ . \tag{6}$$

$$P[C_0,A_0] = Jx_1[c_1,a_1] \times ... \times Jx_t[c_t,a_t] \times Jx_{t+1}[d_{(t+1),0},b_{(t+1),0}] \times ... \times Jx_n[d_{n0},b_{n0}] \ . \tag{7}$$

$$P[C_1,A_1] = Jx_1[c_1,a_1] \times ... \times Jx_t[c_t,a_t] \times Jx_{t+1}[d_{(t+1),1},b_{(t+1),1}] \times ... \times Jx_n[d_{n1},b_{n1}] \ . \tag{8}$$

Let's mark $w_0 = F(B_0)$, $e_0 = F(D_0)$, $w_1 = F(B_1)$, $e_1 = F(D_1)$.

Assume $w_0 = w_1 = w$ and $e_0 = e_1 = e$, so the products of (5) and (6) belong to the same subset of $\boldsymbol{K(e,w)}$. Considering (1) for any corteges that are used alongside forming products from $\boldsymbol{Q}(\hat{C}, \hat{A},e,w)$ we have

$$f(A_0) = f_1(\hat{A},w) = f(A_1) \ . \tag{9}$$

$$f(C_0) = f_1(\hat{C},e) = f(C_1) \ . \tag{10}$$

Let's examine the case when $e \neq w$. Then $F(B_0) \neq F(D_0)$, $F(B_1) \neq F(D_1)$, i.e. $B_0 \neq D_0$, $B_1 \neq D_1$, and thus $A_0 \neq C_0$, $A_1 \neq C_1$ independently of equality or inequality of the $\hat{A}$ and $\hat{C}$ corteges. From (9) and (10) it comes out that the products (7) and (8) belong to the same sets separation class of $\boldsymbol{H_{(1,n)}}$. Indeed, if $f(A_0) = f(C_0)$, then this is one of the classes of $\boldsymbol{K^0_c(f)}$ or $\boldsymbol{K^1_c(f)}$ self-correction, if $f(A_0) \neq f(C_0)$, then this is one of the distorted f – $\boldsymbol{K^0_e(f)}$ or $\boldsymbol{K^1_e(f)}$ functions classes.

Let's examine the case when $e = w$ (case of the F functions' cortege self-correction). In this case two options are possible.

Option 1.

$\hat{A} \neq \hat{C}$, then $A_0 \neq C_0$, $A_1 \neq C_1$ independently of the correlation between B_0 and D_0 and between B_1 and D_1 . As before, the products (7) and (8) belong to the same sets separation class of $\boldsymbol{H_{(1,n)}}$ or the class of self-correction or the class of distorted values of function f .

Option 2.

$\hat{A} = \hat{C}$, $B_0 \neq D_0$, $B_1 = D_1$. In this case $A_0 \neq C_0$, $A_1 = C_1$. We have $f(A_0) = f_1(\hat{A},w) = = f(A_1)$, $f(C_0) = f_1(\hat{C},w) = f(C_1)$.

Since $\hat{A} = \hat{C}$, than $f(A_1) = f(C_1) = f(A_0) = f(C_0)$, then the product (7) will belong to one of the $\boldsymbol{K^0_c(f)}$ or $\boldsymbol{K^1_c(f)}$ classes, and the product (8) is to one of the $\boldsymbol{K^0_w(f)}$ or $\boldsymbol{K^1_w(f)}$ classes, i.e. $\boldsymbol{Q}(\hat{C}, \hat{A},e,w) \subset \boldsymbol{K^0_w(f)} \cup \boldsymbol{K^0_c(f)}$, or $\boldsymbol{Q}(\hat{C}, \hat{A},e,w) \subset \boldsymbol{K^1_w(f)} \cup \boldsymbol{K^1_c(f)}$.

Option 3.

$\hat{A} = \hat{C}$, $B_0 = D_0$, $B_1 \neq D_1$, $A_0 = C_0$, $A_1 \neq C_1$. This option is identical to the Option 2.

Option 4.

$\hat{A} = \hat{C}$, $B_0 \neq D_0$, $B_1 \neq D_1$, $A_0 \neq C_0$, $A_1 \neq C_1$. This option is identical to the Option 3.

Application of the outlined technique can substantially reduce the complexity of the self-correction properties of the Boolean functions. For instance, given $n=8$, and different values of t and r, the acceleration by several times in calculations is being observed, given the n=16 - several thousand times.

2.2 Using the properties of Boolean functions

For a random Boolean function of $f(x_1, x_2,\ldots,x_n)$, according to the previous, we shall determine the set $M(f)$ of all the $A_j = <a_{1j},a_{2j},\ldots,a_{nj}>$, $j=0,1,\ldots,2^n-1$ corteges, determine two-dimensional array $GX_{(1,n)}[C, A]$, where any element $GX_{(1,n)}[C_l, A_j]$ will contain the product value of $P[C_j, A_j]$ (2) and two-dimensional array $Jf[0..1,0..1]$.

In practice it is believed that the indubitably correct function' values are determined as a probability of false argument function values absence. We shall mark this probability as $P(f)$, it is equal to the sum of diagonal elements of $GX_{(1,n)}[A,C]$ array. In fact, the indubitably correct function values are calculated as the sum of $P_c(f) = Gf[0,0]+Gf[1,1]$. We shall mark the difference between the given values as $P_a(f)$ is the possibility of self-correction. In the given context the complexity of self-correction values calculation equals to the complexity of all $GX[C,A]$ array elements calculation, i.e. $4^n(n-1)$ floating-point multiplication operations.

We shall split the $f(x_1, x_2,\ldots,x_n)$ function variables manifold into k groups. We shall mark m_i (i=1,2,…,k) as variables quantity in the group, $m_1+m_2+\ldots+m_k=n$. We shall calculate the products in every variables group separately, representing the 4^{mi} **X** (m_i- -1) multiplication operations for the i-group. Using the products multiplications against each other, we shall receive the $H_{(1,n)}$ products set. The total quantity of operations will be equal to:

$$4^{m_1} \times (m_1 - 1) + 4^{m_2} (m_2 - 1) + \ldots + 4^{m_k} \times (m_k - 1) + 4^n \times (k - 1) \leq 4^n \times (n - 1). \qquad (11)$$

The maximal value (4^n **X** $(n-1)$) of the correlation' left side (11) is achieved upon $m=1$, i.e. when the separation into groups is absent. The minimal value is achieved upon k=2 and $m_i=n/2$ and equals to $2 \times 4^{n/2} + 4^n$. Herewith, the multiplication operations quantity diminishes approximately $(n-1)$ times. For the $f(x_1, x_2,\ldots,x_n)$ function we shall select the following option to separate arguments into two groups ($x_1, x_2,\ldots,x_m$) and ($x_{m+1}, x_{m+2},\ldots,x_n$). We shall create the $GX_{(1,m)}$ array of all the products of (2) kind for the first group of values and $GX_{(m+1,n)}$ array for the second group. According to (11), multiplication operations quantity will be $4^m \times (m - 1) + 4^{n-m} \times (m - n - 1) + 4^n$.

Further reduction of this quantity can be achieved taking into account the properties of $f(x_1, x_2,\ldots,x_n)$ function. Using the enumeration of all the possible variables of the first group we shall create 2^m derived $f(d_1, d_2,\ldots,d_m, x_{m+1}, x_{m+2},\ldots,x_n)$ functions. Assume s as the quantity of different derived($1 \leq s \leq 2^m$) functions. Using r let's mark the quantity of different derived $f(x_1, x_2,\ldots,x_m, d_{m+1}, d_{m+2},\ldots,d_n)$. Then, the computational complexity will be:

$$4^m \times (m - 1) + 4^{n-m} \times (m - n - 1) + s^2 \times r^2 \quad . \qquad (12)$$

For instance,

$$f(0, 0,\ldots,0, x_{m+1}, x_{m+2},\ldots,x_n) = f(0, 0,\ldots,1, x_{m+1}, x_{m+2},\ldots,x_n). \qquad (13)$$

From the $GX_{(m+1,n)}$ we shall select a random product of $Gx_{m+1}[c_{m+1}, a_{m+1}] \times \ldots \times Gx_{m+1}[c_n, a_n]$. Creating the $H_{(1,n)}$ we will have the two following products:

$$(Gx_1[0, 0] \times \ldots \times Gx_m[0, 0]) \times (Gx_{m+1}[c_{m+1}, a_{m+1}] \times \ldots \times Gx_{m+1}[c_n, a_n]). \qquad (14)$$

$$(Gx_1[0, 0] \times \ldots \times Gx_m[0, 1]) \times (Gx_{m+1}[c_{m+1}, a_{m+1}] \times \ldots \times Gx_{m+1}[c_n, a_n]). \qquad (15)$$

Since the condition (13) is met upon any $x_{m+1}, x_{m+2}, \ldots, x_n$ variables values ten the products (14) will belong to the same $H_{(1,n)}$ set class and will be used in the corresponding products' sum to determine the Jf array. The products located in line and column of the $x_1, x_2, \ldots, x_m$ variables with matching derived functions will be located in the same separation class of the $H_{(1,n)}$ set (see Table 1 for the details).

Table 1. Uniting the columns on non-distorted, lines on distorted variables values.

C\A	00..00	…	<$a_1 \ldots a_m$>	…	11..11
00..00	P[00..00,00..00]+ P[00..00,00..01]+ P[00..01,00..00]+ P[00..01,00..01]	…	P[00..00,$a_1..a_m$]+ P[00..01,$a_1..a_m$]		P[00..00,11..11]+ P[00..01,11..11]
…	…	…	…	…	…
<$c_1 \ldots c_m$>	P[$c_1..c_m$,00..00]+ P[$c_1..c_m$,00..01]	…	P[$c_1..c_m$, $a_1..a_m$]	…	P[$c_1..c_m$, 11..11]
…	…	…	…	…	…
11..11	P[11..11,00..00]+ P[11..11,00..01]	…	P[11..11, $a_1..a_m$]	…	P[11..11,11..11]

The examples given are valid for any $x_1, x_2, \ldots, x_m$ variables values, that result in identical derived functions. The same way, a proof can be made for the second group of variables and derived functions. The final proof of validity (3) is based on the distributive law and symmetry of addition and multiplication operations.

$$\delta = \frac{4^n (n-1)}{4^m \times (m-1) + 4^{n-m} \times (m-n-1) + s^2 \times r^2} \ . \qquad (16)$$

The correlation (16) shows many times the multiplication operation of the techique proposed differs from the standard one.

3 Methods for devices optimization

3.1 Optimization of pre-definitions

Quite often in the engineering practice the Boolean functions of $F(x_1,x_2,\ldots,x_n)$ cortege are not determined on all the codes of $\boldsymbol{M(F)}$ set of argument values. Implementing the combinational scheme, the Boolean functions are pre-defined in order to optimize scheme based on some parameters. For the parameter, we shall choose the self-correction parameter.

We shall mark $\boldsymbol{W(F)} \subset \boldsymbol{M(F)}$ as a set of codes, on which the functioning of combinational scheme is pre-defined under the technical requirements, $\boldsymbol{E(F) = M(F) / W(F)}$. Assume in the result of distortions from the $\boldsymbol{W(F)}$ set' codes the $\boldsymbol{M(F)}$ codes will be created with some probability. We shall determine the $<D_u, B_s, P_{DB}, j>$ corteges, where $B_s \in \boldsymbol{W(F)}_r$, D_u is the result of B_s code distortion, P_{DB} is a probability of such a distortion, $j = F(B_s)$ are codes on the combinational scheme outputs. In the real practice the probability of codes distortion from $\boldsymbol{W(F)}$ to the codes from $\boldsymbol{W(F)}$ can be unequal.

We shall mark Z as a set of $<D_u, B_s, P_{DB}, j>$ corteges, implying that any B_s code from $\boldsymbol{W(F)}$ with one or another probability can be distorted into any code from $\boldsymbol{M(F)}$. For the pre-determination of Boolean functions of the $F(x_1,x_2,\ldots,x_n)$ cortege, in order to ensure the maximal self-correction indicator, the following algorithm is proposed:

Algorithm 1.

1). Delete the corteges containing $D_u \in W(F)$ from Z-set. According to technical requirements, the D_u distortions are not sensitive, if $D_u \in K_v$ (self-correction), or

the functions' cortege forms the false value. In the result such removal a Z_1 set will be formed, where in every cortege $D_u \in E(F)$.

2). Split the Z_1 set into $Z_1(D_u)$ subsets where in the corteges the values of D_u codes match.

3). In every set $Z_1(D_u)$ find the cortege with maximal value of P_{DB} and pre-determine the functions' cortege on a D_u code with a V code value of this cortege.

Given the distortion probabilities, a received pre-determination provides the maximal self-correction value. Indeed, a D_u code can be formed in the result of several codes distortion from $W(F)$. Choice of the most probable distortion provides the bigger probability value of self-correction.

3.2 Optimization with structural synthesis of machines

Usually in engineering practice the technical requirements for implementation of combinational scheme are set as $y=f(x)$ function, where $x \in A=\{a_1,a_2,\ldots,a_w\}$, $y \in B=\{b_1,b_2,\ldots,b_u\}$ are finite sets of random objects. At the same time, the problem of optimal encoding by the binary codes of elements of A and B sets on different criteria is being solved. Let's examine encoding optimization based on criteria of self-correction indicators. We shall separate the A set into classes of K_v non-sensitivity, where any a_{i1}, a_{i2} of elements from A belong to K_v, if $f(a_{i1})= f(a_{i2}) = b_v$ $(v = 1,2,\ldots,u)$. It is obvious, that the encoding of B-set symbols does not affect indicators of self-correction. To determine optimal elements encoding of the A-set it is sufficient to calculate probabilities sum of mutual distortions within the classes given and choose encoding with maximal sum. Totally, there are $N=(2^n(2^n-1)(2^n-2)\ldots(2^n-w+1)$ encodings possible, where n is codes' number of digits. For example, even with $n=6$ and $w=50$ $N>10^{78}$. That is why the following heuristic algorithm is proposed. Assume that for the input data an array $GX(1,n)[C,A]$, where any element $GX(1,n)[C_l, A_j]$ $(l,j \in \{0,1,\ldots,2^n-1\})$ contain value of the $P[C_l,A_j]$ product (2). Based on this array, a GXT auxiliary array is created, that contains C_l,A_j codes and value of $P[C_l,A_j] + P[A_j,C_l]$ sum with $A_j \neq C_l$. All elements of the GXT array are listed as non-marked.

Algorithm 2.

1). Choose the foremost K_v class with the biggest value of $\#K_v$.

2). Create a TK_v array from all combinations of two elements of the class. All elements of the TK_v array consider as non-marked.

3). If all elements of the TK_v array array are marked, then go to point 1.

4). Choose the foremost pair (i.e. a_{i1}, a_{i2}) from the TK_v array.

5). Choose from the GXT array the foremost non-marked element with the biggest value of probabilities sum. Assign $a_{i1} = C_l$, $a_{i2} = A_j$, where the value (C_l,A_j) correspond to the sum chosen. Assign $c = C_l$, $a = A_j$. Mark up the chosen element in the GXT array. Mark up the (a_{i1}, a_{i2}) pair in the GXT array.

6). Choose form the TK_v array the foremost non-marked pair, that contains a_{i1} or a_{i2} elements. Choose from the GXT structures array the foremost non-marked element with maximal sum under condition that one of values of one of the codes of the chosen sum equals to already determined (c or a). For the second element of pair, for instance (a_{i3}) the second from codes chosen is assigned. Mark up pairs (a_{i1}, a_{i3}) and (a_{i2}, a_{i3}) of the TK_v array and the corresponding structures of the GXT array. Repeat Point 6 until all pairs containing a_{i1} or a_{i2} elements will not be marked up.

7). If the K_v class contains only one element, then assign it any code from non-used previously.

8). If there are classes non-worked out, go to Point 1.

The example examined does not guarantee optimal solution, but provides the result close enough to the optimal. In case when $\#A$ is not the power of 2, for the pre-determination of Boolean functions it is necessary to use Algorithm 1.

4 Conclusions

The Statement proven enables the practical possibility to calculate self-correction parameters considering the real devices structure, in which always the multiple-digits devices are implemented as the composition of lesser digits-quantity' devices. In separate cases a method can be used that is based on the Boolean functions properties, and not on the actual implementation.

The obtained results are a basis for analysis and experimental checkup of the devices optimization methods based on self-correction parameter upon condition of incomplete determinacy of Boolean functions or when to determine Boolean functions it is required to carry out the encoding stage of input alphabet elements. According to point 3 of the Algorithm 1 and points 5 and 6 of the Algorithm 2, upon the equality of probabilities, a choice can be ambiguous. This enables for carrying out the optimization based on other parameters.

References

8. Klyatchenko, Y., Tarasenko, V., Tarasenko-Klyatchenko, O., Teslenko, O. : Reliability evaluation method of functioning logic networks in the distortion of input data determined. Radioelectronics and Informatics 5, 165-169 (2014).
9. Klyatchenko, Y.: The reliability determination of the hardware devices on FPGAs functioning under conditions of input logic signals distortion. Informational technologies and computer engineering 3, 9-12 (2015).
10. Klyatchenko, Y., Tarasenko, V., Tarasenko-Klyatchenko, O., Teslenko, O.: The probability of correct operation for logic networks in condition of input signals distortion. Computer-integrated technologies: education, science, production 8, 47-52 (2012).
11. Afshord, S.T., Pottosin, Y.: Improved Decomposition for a System of Completely Specified Boolean Functions. IJITCS 6 (1), 25-32 (2014). DOI: 10.5815/ijitcs.2014.01.03
12. Ritika Wason, R., Soni, A. K. , Qasim Rafiq, M.: Estimating Software Reliability by Monitoring Software Execution through OpCode. IJITCS 7(9), 23-30 (2015). DOI: 10.5815/ijitcs.2015.09.04
13. Vijayakumari, C. K, Mythili, P., Rekha, K. James: A Simplified Efficient Technique for the Design of Combinational Logic Circuits. IJISA 7(9), 42-48 (2015). DOI: 10.5815/ijisa.2015.09.06
14. Georgiev, D., Tentov, A.: FSM Circuits Design for Approximate String Matching in Hardware Based Network Intrusion Detection Systems. IJITCS 6(1), 68-75 (2014). DOI: 10.5815/ijitcs.2014.01.08

УДК 004.6

К.т.н., доцент Клятченко Я.М., студент Тарасенко Г.О.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

АПАРАТНІ СТРУКТУРИ ДЛЯ ПОРІВНЯННЯ ЧИСЕЛ В НЕГАПОЗИЦІЙНІЙ СИСТЕМІ ЧИСЛЕННЯ

Abstract

Yaroslav Klyatchenko, assoc. prof., PhD; Georgii Tarasenko, student

Hardware structures for numbers comparison in negative-base numeral system

The paper is dedicated to analysis of the opportunities to use negative-base numeral system base -2 to build arithmetical expanders of computer systems based on microprocessors. It is shown, that negative-base numeral system has specific advantages, in comparison with binary positional system.

Вступ

Застосування арифметичних розширювачів є дієвим способом забезпечення підвищеної продуктивності інструментальних засобів інформаційних технологій, виконаних на основі мікропроцесорів. Розробка арифметичних розширювачів дає змогу провести апробацію нових нетрадиційних підходів до їх структурної організації і до створення їх алгоритмічного забезпечення, а також нових принципів подання чисел і використання нетрадиційних систем числення. Відомо багато позиційних систем

числення, які можуть використовуватися у комп'ютерних системах для автоматизації обчислень. Кожна із них має свої переваги та недоліки, які і визначають можливості їх конкретного використання. Подання чисел в нега-позиційних (тобто, позиційних з від'ємною основою) системах числення є однією із цікавих можливістей покращити структурно-функціональні та техніко-економічні характеристики комп'ютерних операційних засобів [1].

Постановка задачі

Вирішення задачі розробки апаратної реалізації порівняння чисел у негапозиційній системі числення є важливим для застосування в спеціалізованих комп'ютерних системах, з огляду на доступність компонентів таких систем.

Теоретичні відомості

Негапозиційна система числення – це позиційна система числення з від'ємною основою. Особливістю таких систем є відсутність знака перед від'ємними числами і, як наслідок, відсутністю правил обробки знаків. Всяке число в будь-якій з негапоизційних систем, що відрізняється від 0, з непарною кількістю розрядів – додатне, з парною – від'ємне [1].

В таких системах числення значно прискорюються найбільш масові операції додавання-віднімання, скільки сигнали переносу (для віднімання сигналу боргу) у випадку використання такого числення, не розповсюджується далі, ніж на один розряд (на одну позицію). Це, в свою чергу, дозволяє отримати час операції додавання-віднімання, який не залежить від довжини операндів, оскільки проблема «довгих ланцюжків» (так звана ‘‘вічна проблема’’ позиційної арифметики взагалі) в такому разі не виникає [1,2]. Іншою заманливою особливістю подання чисел в позиційних численнях з від'ємною основою є відсутність спеціальної знакової позиції (розряду). Але слід мати на увазі, що переваги подання чисел в негапозиційних численнях нівелюються складністю виконання інших операцій з таким поданням чисел. Основною мірою це стосується

операцій порівняння. Відомі навіть рекомендації щодо попереднього переводу чисел до зміщених систем числення заради необхідності проведення порівняння.

Опис алгоритму

З метою спростування сумнівів щодо можливості технічного відтворення операції порівняння чисел в негапозиційному численні пропонуються деякі алгоритмічні та структурні рішення, які є цілком реальними з позицій досягнень сучасної комп'ютерної схемотехніки.

Нехай A і B – двійкові операнди в позиційній системі числення з основою *k=-2,* причому:

$$A = \sum_{i=0}^{n} a_i (-2)^i; B = \sum_{i=0}^{n} b_i [(-2]^i); a_i, b_i \in \{0,1\}; i = \overline{0,n};$$

$$i = \overline{0,n}; n = 2l$$

Тут *n*-число всіх розрядів для подання операндів, яке без втрати загальності викладу будемо вважати парним; *l* – число розрядів в поданні операндів *A* і *B*, які мають один і той же знак їх ваги (інакше – число розрядів з додатною і від'ємною вагою однакове і дорівнює *l*). Очевидно, що за прийнятих позначень максимальний операнд в такому негапозиційному численні може бути обчислений як:

$$A_{max} = 2^0 + 2^2 + 2^4 + \cdots + 2^{n-2} = \sum_{i=0}^{n-1} 2^{2i},$$

а найменший операнд, як:

$$A_{min} = -\left(2^1 + 2^3 + 2^5 + \cdots + 2^{n-1}\right) = -\sum_{i=1}^{n-1} 2^{2i-1}$$

Наприклад, якщо *n=8, l=4,* то ваговий ряд для такої негапозиційної системи має вигляд *-128, 64, -32, 16, -8, 4, -2, 1*, а діапазон можливих цілих

чисел знаходиться між $\mathbf{A}_{\max}$=85 і $\mathbf{A}_{\min}$=170. Реалізація порівняння операндів *A* і *B* означає встановлення факту виконання одного із співвідношень: *A=B, A>B, A<B*.

Основними властивостями подання операндів в такій негапозиційній системі, що забезпечують досягнення поставленої мети, є наступні:

1. Знаки ваги окремих розрядів чергуються оскільки $(-2)^i > 0$, якщо *i* – парне (включаючи випадок *i*=0) та $(-2)^i < 0$, якщо *i*-непарне. Це дозволяє розділити кожен окремий операнд на додатну і від'ємну частину та виконувати операції з кожною частиною окремо.

2. Будь-яка ненульова цифра, що знаходиться на *i*-му місці (рахуючи справа наліво) в запису операнда за абсолютною величиною її кількісного еквівалента перевищує кількісний еквівалент будь-якої комбінації цифр, що знаходиться справа від *i*-ї позиції. Ця властивість дозволяє в багатьох випадках проводити порівняння чисел орієнтуючись на позиції крайніх зліва одиниць в поданні операнда.

3. Знак операнда в цілому або деякої його частини визначається знаком ваги найстаршого ненульового розряду. Отже якщо група із *m* старших розрядів операнда *A* тотожна (збігається) групі із *m* старших розрядів операнда *B*, то їх можна виключити із розгляду, а остаточний висновок про співвідношення *A* і *B*, робити по *n-m* – розрядних фрагментах *A* і *B*, які залишаються.

Таким чином, алгоритм порівняння операндів *A* і *B*, поданих в негапозиційному численні, полягає в наступному:

1. На основі двійкових наборів (векторів), що відповідають операндам *A* і *B*, утворити вектор *C* шляхом їх покомпонентного додавання за модулем 2. Якщо всі компоненти вектора *C*

дорівнюють нулю, то $A=B$ і операція порівняння закінчена. В іншому разі перейти до п.2.

2. Шляхом пріоритетного порівняння компонент вектора C утворити маркерний вектор D, який містить лише одну одиницю в позиції d_M, що відповідає найстаршій одиниці вектора C. Це означає, що в векторах A і B всі цифри лівіше позиції d_M не впливають на виконання співвідношення порівняння.

3. Шляхом покомпонентного порівняння в парах $[\![(a]\!]_i, d_i)$ та $[\![(b]\!]_i, d_i)$ визначити цифри $a_M^{'}$ і $b_M^{'}$ в векторах A і B, які знаходяться на позиції маркерного розряду d_M в векторі D.

4. Визначити парність (чи непарність) позиції маркерного розряду d_M у векторі D. Позначимо $P(d_M) = 1$ – факт парності d_m, $N(d_M) = 1$ – факт непарності d_m. Очевидно, що $P(d_M)$ та $N(d_M)$ можуть бути тільки взаємно інверсними.

5. На основі значень $a_M^{'}$, $b_M^{'}$,, відповідно до таблиці 1 сформувати результат порівняння чисел A і B. Алгоритм закінчено.

Таблиця 1.

$a_M^{'}$	$b_M^{'}$	$P(d_M)$		Співвідношення між операндами
1	0	1	0	$A>B$
0	1	0	1	$A>B$
1	0	0	1	$A<B$
0	1	1	0	$A<B$
0	0	0	0	$A=B$

Апаратна реалізація такого алгоритму не становить труднощів, оскільки використовуються компоненти сучасних ПЛІС, що є цілком реальним для ефективної реалізації пропонованого алгоритму [3,4].

Висновки

Аналіз алгоритму показує, що всі структурні елементи, які використані для встановлення співвідношень порівняння операндів у нега-позиційному численні, можуть бути реалізовані комбінаційними схемами, що функціонують у двозначному структурному алфавіті. Очевидно також, що всі вони можуть бути відтворені на базі стандартних логічних ресурсів сучасних програмовних логічних інтегральних середовищ (ПЛІС). Отже, реалізація порівняння операндів в нега-позиційному численні є принципово можливою і здійсненною без переводу операндів до позиційної системи числення. Стосовно структури слід зауважити, що оптимізація апаратних засобів, використовуваних на деяких кроках алгоритма, наприклад, заміна комбінаційного формувача парності на лічильники, може породити нові ефективніші реалізації.

Література

1. Корнійчук В. І., Тарасенко В. П., Тарасенко-Клятченко О. В. Основи комп'ютерної арифметики. В. І. Корнійчук, В. П. Тарасенко, О. В. Тарасенко-Клятченко. –К.: «Корнійчук», 2014, -с.222.
2. Клятченко Я. М., Тарасенко Г. О., Тарасенко-Клятченко О. В. Реалізація подання чисел в негапозиційних системах. Я. М. Клятченко. Г. О. Тарасенко., О. В. Тарасенко-Клятченко. Збірник доповідей МНТК «Програмовані логічні інтегральні схеми та мікропроцесорна техніка в освіті і виробництві». Видавництво ЛНТУ, місто Луцьк, -2016, -с.16…20
3. Использование прогорамируемых логических интегральных схем (ПЛИС) (Электронны учебник по курсу компьютерная электроника, Санкт-Петербургского Института точной механики и оптики)
4. Кнышев Д.А., Кузелин М.О. ПЛИС фирмы Xilinx: описание структуры основных семейств. М.: издательский дом «Додэка-XXI», 2001.